

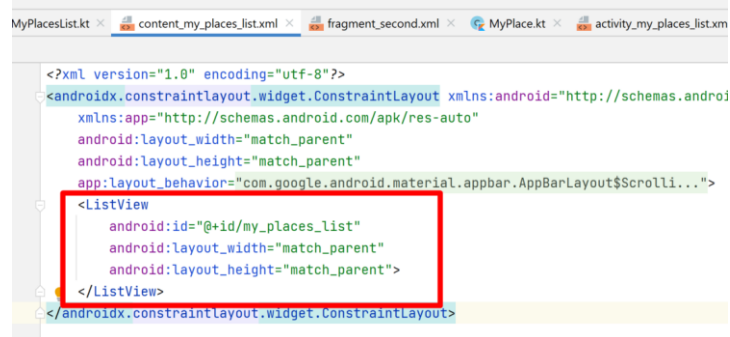
## MOSIS - Laboratorijska vežba 3

Na prošloj vežbi je kreiran osnovni kostur aplikacije *MyPlaces*. Osnovni cilj vežbe je bio upoznavanje sa osnovnim gradivnim elementima Android aplikacije i kako je moguće dodavati nove Activity-e u Android aplikaciju. Od najave koju je Android tim imao 2018. koncept Android aplikacija se menja i fokus prelazi na takozvane *SingleActivityApplication* aplikacije. Takve aplikacije podrazumevaju jedan *Activity* kao ulaznu tačku aplikacije koja dalje kontroliše podatke prikazane korisniku preko jednostavnijih aplikacionih odredišta koji sadrže komponente korisničkog interfejsa. Opciono, aplikacija može imati još neku aktivnost ali je osnovni interfejs aplikacije implementiran u jednoj aktivnosti. Kao osnovna odredišta aplikacija ustalili su se fragmenti (*Fragments*). Nakon što je Android ekosistem proširen *Jetpack* bibliotekom, postalo je moguće da se za jedan *Activity* definiše više fragmenata i opiše navigacija između njih. To je omogućilo korišćenje većeg broja fragmenata odjednom, ukoliko veličina ekrana to dozvoljava (npr. na tabletima) kao i ponovno korišćenje fragmenata u drugim aktivnostima aplikacije.

Do prelaska na *SingleActivityAplikaciju*, deljenje podataka između različitih aktivnosti aplikacije je bilo potrebno vršiti pomoću instanci klasa, koje su najčešće implementirali *Singleton* pattern, i koje su imale aplikacioni kontekst i doseg (*Application Scope*) imale su životni vek procesa aplikacije koja je startovana. Zbog toga, tim podacima su mogli pristupiti i servisi i *Content Provider*-i a ne samo aktivnosti u kojima je bilo potrebno deliti podatke. Zbog toga, u modernom pristupu razvoju Android aplikacija, deljenje podataka između različitih aktivnosti nije poželjno već je ohrabren prelazak na arhitekturu aplikacije zasnovane na jednoj aktivnosti i više fragmenata koji pokrivaju različite ekrane aplikacije. Još jedna prednost novog pristupa je i bolja tranzicija između dva ekrana, ostvarena korišćenjem fragmenata za razliku od korišćenja aktivnosti.

Ciljevi ove vežbe su:

1. Prebacivanje postojeće aplikacije u *SingleActivityApplication* arhitekturu.
  2. Kreiranje modela omiljenog mesta
  3. Kreiranje deljene *ViewModel* klase koja će držati podatke o omiljenim mestima a koja će biti deljena između fragmenata aplikacije.
  4. Dodavanje fragmenta za unos nove omiljene lokacije.
1. Prebaciti postojeću aktivnost *MyPlacesList.kt* u *SecondFragment* prve aktivnosti *MainActivity*.
    - a. Iz fajla *content\_my\_places\_list.xml* iskopirati deo u kome je definisan *Listview*.



- b. U fajlu `fragment_second.xml`, obrisati postojeći `TextView` i `Button` i nalepiti iskopirani `ListView`.
- c. U klasi `SecondFragment.kt` obrisati postavljanje `ClickListener`-a za dugme `buttonSecond`.

```

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

    binding.buttonSecond.setOnClickListener {
        findNavController().navigate(R.id.action_SecondFragment_to_FirstFragment)
    }
}

```

- d. U klasu `SecondFragment.kt` u metodu `onViewCreated` dodati kod iz klase `MyPlacesList.kt` koji inicijalizuje listu.

```

class MyPlacesList : AppCompatActivity() {

    private lateinit var appBarConfiguration: AppBarConfiguration
    private lateinit var binding: ActivityMyPlacesListBinding
    private lateinit var places: ArrayList<String>

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        binding = ActivityMyPlacesListBinding.inflate(layoutInflater)
        setContentView(binding.root)
        setSupportActionBar(binding.toolbar)

        supportActionBar?.setDisplayHomeAsUpEnabled(true)

        places = ArrayList<String>()
        places.add("Tvrdjava")
        places.add("Cair")
        places.add("Park Svetog Save")
        places.add("Trg Kralja Milana")
        val myPlacesList: ListView = findViewById<ListView>(R.id.my_places_list)
        myPlacesList.adapter = ArrayAdapter<String>(context: this, android.R.layout.simple_list_item_1, places)

        binding.fab.setOnClickListener { view ->
            Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)
                .setAction("Action", null).show()
        }
    }
}

```

- e. Uvesti paket u kome je definisan `ArrayAdapter` (korišćenjem `Alt+Enter` dok je kursor na njemu) Potrebno je ispraviti liniju u kojoj se pribavlja `ListView` i izmeniti način kako se pribavlja kontekst za koji se `ArrayAdapter` kreira.

```

// This property is only valid between onCreateView and
// onDestroyView.
private val binding get() = _binding!!
private lateinit var places: ArrayList<String>

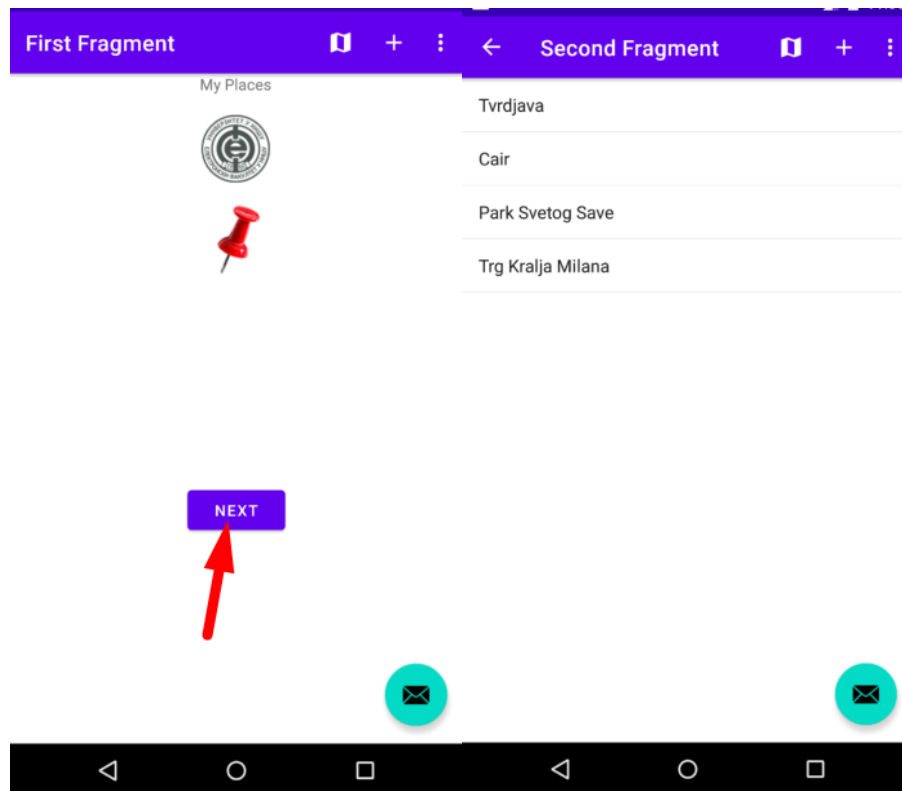
override fun onCreateView(
    inflater: LayoutInflater, container: ViewGroup?,
    savedInstanceState: Bundle?
): View? {
    _binding = FragmentSecondBinding.inflate(inflater, container, attachToParent: false)
    return binding.root
}

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

    places = ArrayList<String>()
    places.add("Tvrdjava")
    places.add("Cair")
    places.add("Park Svetog Save")
    places.add("Trg Kralja Milana")
    val myPlacesList: ListView = requireView().findViewById<ListView>(R.id.my_places_list)
    myPlacesList.adapter = ArrayAdapter<String>(view.context, android.R.layout.simple_list_item_1, places)
}

```

- f. Probati aplikaciju. Nakon klika na dugme Next prvog fragmenta, drugi fragment bi trebalo da učitava statičku listu.



- g. Obrisati klasu `MyPlaceList.kt` i njene povezane fajlove za UI ( `layout/activity_my_places_list.xml` i `layout/content_my_places_list.xml` ) a zatim u klasi `MainActivity.kt` u `onOptionsItemSelected` zameniti poziv za startovanje `MyPlacesActivity`, pozivom za navigaciju na drugi fragment (iskopirati iz prvog fragmenta poziv koji izvršava dugme `Next`).
- h. Pošto pokušavamo da iz aktivnosti koja sadrži fragment menjamo fragmente korišćenjem fragment navigatora, kontrole koja upravlja tranzicijama između fragmenata, potrebno je proslediti id fragment navigator komponente. To je moguće naći u `content_main.xml` fajlu tekuće aktivnosti.

```

fragment_second.xml × content_main.xml × MainActivity.kt × fragment_first.xml × nav_graph.xml
1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res-auto"
3     xmlns:app="http://schemas.android.com/apk/res-auto"
4     android:layout_width="match_parent"
5     android:layout_height="match_parent"
6     app:layout_behavior="com.google.android.material.appbar.AppBarLayout$Scroll
7
8 <fragment
9     android:id="@+id/nav_host_fragment_content_main"
10    android:name="androidx.navigation.fragment.NavHostFragment"
11    android:layout_width="0dp"
12    android:layout_height="0dp"
13    app:defaultNavHost="true"

```

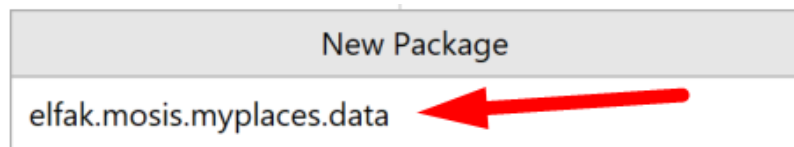
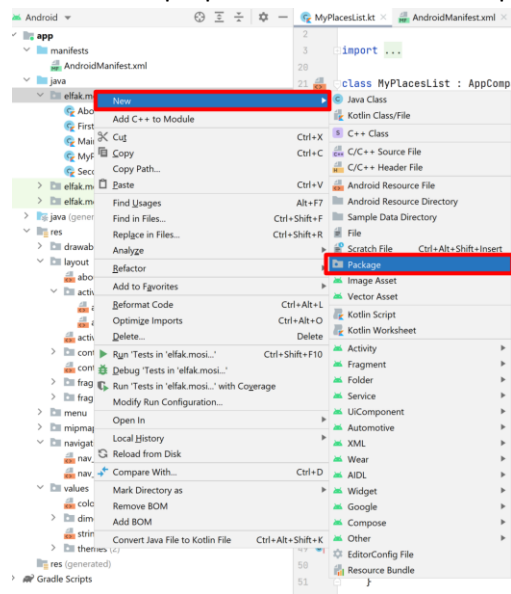
- i. Izmenjeni kod u `MainActivity.kt` klasi treba da izgleda kao na slici.

```

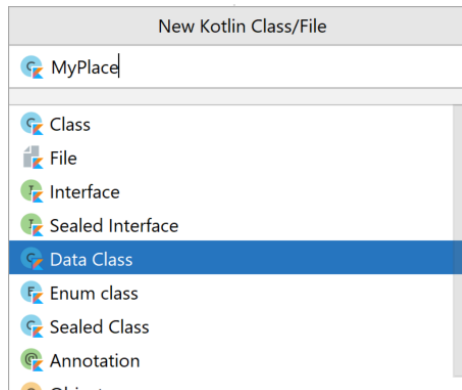
override fun onOptionsItemSelected(item: MenuItem): Boolean {
    // Handle action bar item clicks here. The action bar will
    // automatically handle clicks on the Home/Up button, so long
    // as you specify a parent activity in AndroidManifest.xml.
    when (item.itemId) {
        R.id.action_show_map -> Toast.makeText(context, this, text: "Show Map!", Toast.LENGTH_SHORT).show()
        R.id.action_new_places -> Toast.makeText(context, this, text: "New Places!", Toast.LENGTH_SHORT).show()
        R.id.action_my_places_list -> {
            this.findNavController(R.id.nav_host_fragment_content_main).navigate(R.id.action_FirstFragment_to_SecondFragment)
        }
        R.id.action_about -> {
            val i: Intent = Intent(context, About::class.java)
            startActivity(i)
        }
    }
    return super.onOptionsItemSelected(item)
}

```

- j. Probati aplikaciju i proveriti da li klik na odgovarajuće dugme *toolbar*-a prebacuje fragment na odgovarajuću listu.
  - k. Obrisati *fragment\_second.xml* (land) bez brisanja ostalih fajlova za *landscape* prikaz.
  - l. Probati aplikaciju!
2. Kreiranje data klase koja opisuje omiljenu lokaciju (*MyPlace.kt*)
    - a. Potrebno je pokrenuti *Android Studio*.
    - b. Kreirati novi potpaket *data* u trenutnom paketu.



- c. U kreiranom paketu napraviti novu *Kotlin data* klasu *MyPlaces.kt* korišćenjem desnog klika na paket u *Android Studio*->*New*->*Kotlin Class-File* a zatim u meniju uneti ime *MyPlaces* i izabrati *Data Class*.



- d. *Data Class* klase omogućavaju jednostavno kreiranje modela podataka koji imaju samo attribute klase. Za takve klase su automatski definisane *getter* i *setter* metode (samo za promenljive, ne i konstante) kao i *toString()*, *equals()*, *hashCode()*, *copy()* metode.
- e. Definisati da konstruktor klase prima dva *String* podatka, *name* i *description* na osnovu kojih će automatski biti definisani atributi klase i odgovarajuće dodatne metode.

```
lyPlacesList.kt x MyPlace.kt x menu_my_places_list.xml x menu_main.xml x
package elfak.mosis.mylaces.data

data class MyPlace(var name:String, var description:String)
|
```

- 3. Kreiranje deljene *ViewModel* klase koja će držati podatke o omiljenim mestima a koja će biti deljena između fragmenata aplikacije.
  - a. *ViewModel* je deo *MVVM* arhitekture kod koje je *ViewModel* klasa zadužena za čuvanje podataka koje će *View* (ili *ViewController*, zapravo *Activity* ili *Fragment*) prikazati u korisničkom interfejsu na osnovu *Model*-a podataka. Na ovaj način se dobija odvajanje zaduženja (*separation of concerns*) različitih delova aplikacije. U ovom slučaju korisnički interfejs ne sadrži podatke koje treba da prikazuje već ih dobija iz *ViewModel* klase. *ViewModel* klasa pribavlja podatke, na osnovu modela podataka, iz nekog izvora podataka, poput baze podataka ili udaljenog servera.
  - b. Kreirati paket *model* i u njemu klasu *MyPlacesViewModel.kt*

```

MyPlacesViewModel
fragment_second.xml × content_main.xml × MyPlacesViewModel.kt ×
package elfak.mosis.mylplaces.model

import androidx.lifecycle.ViewModel

class MyPlacesViewModel: ViewModel()

```

- c. Dodati atribut klase koji će držati listu omiljenih lokacija (prekopirati iz FragmentSecond.kt)

```

import androidx.lifecycle.ViewModel
import elfak.mosis.mylplaces.data.MyPlace

class MyPlacesViewModel: ViewModel() {
    var myPlacesList: ArrayList<String> = ArrayList<String>()
}

```

- d. Obrisati instanciranje liste iz FragmentSecond.kt klase a zatim dodati kreiranje instance kreirane *MyPlacesViewModel* klase. Ovo kreiranje se vrši delegiranjem nadklasi (ViewModel) kreiranog *MyPlacesViewModel*-a. To zapravo znači da će osnovni get i set metode atributa *MyPlacesViewModel* klase biti kontrolisani od strane ugrađenog viewModel delegata čime se praktično sve reakcije na promene podatka obrađuju preko njega.

```

SecondFragment.kt
/**
 * A simple [Fragment] subclass as the second destination in the navigation.
 */
class SecondFragment : Fragment() {

    private var _binding: FragmentSecondBinding? = null

    // This property is only valid between onCreateView and
    // onDestroyView.
    private val binding get() = _binding!!
    private val myPlacesViewModel: MyPlacesViewModel by viewModel()
    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        _binding = FragmentSecondBinding.inflate(inflater, container, attachToParent: false)
        return binding.root
    }
}

```

- e. A zatim zameniti i pristup listi tako da joj se pristupa preko *viewmodel*-a.

```

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

    myPlacesViewModel.myPlacesList.add("Tvrdjava")
    myPlacesViewModel.myPlacesList.add("Cair")
    myPlacesViewModel.myPlacesList.add("Park Svetog Save")
    myPlacesViewModel.myPlacesList.add("Trg Kralja Milana")
    val myPlacesList: ListView = requireView().findViewById<ListView>(R.id.my_places_list)
    myPlacesList.adapter = ArrayAdapter<String>(view.context, android.R.layout.simple_list_item_1, myPlacesViewModel.myPlacesList)
}

```

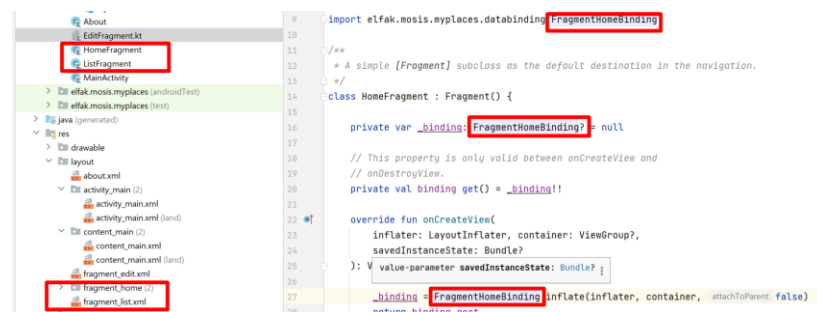
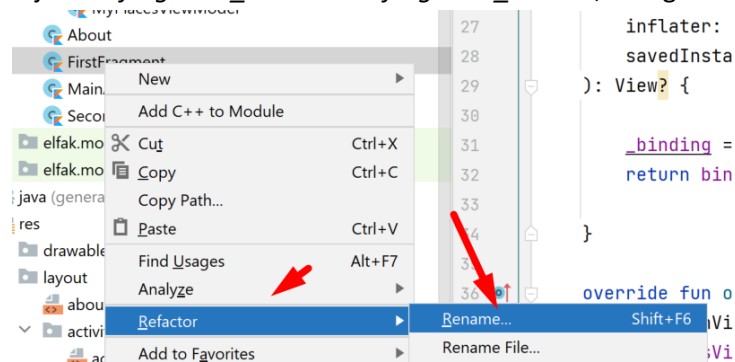
- f. Probati aplikaciju!
- g. Dodati *addPlace* metodu u klasu *MyPlacesViewModel.kt* klasu a zatim zameniti dodavanje novih lokacija tako da koriste dodatnu metodu.

```
import androidx.lifecycle.ViewModel
import elfak.mosis.myplaces.data.MyPlace

class MyPlacesViewModel: ViewModel() {
    var myPlacesList: ArrayList<String> = ArrayList<String>()
    fun addPlace(place: String){
        myPlacesList.add(place);
    }
}

override fun onCreateView(view: View, savedInstanceState: Bundle?) {
    super.onCreateView(view, savedInstanceState)
    myPlacesViewModel.addPlace( place: "Tvrdjava")
    myPlacesViewModel.addPlace( place: "Cair")
    myPlacesViewModel.addPlace( place: "Park Svetog Save")
    myPlacesViewModel.addPlace( place: "Trg Kralja Milana")
    val myPlacesList: ListView = requireView().findViewById<ListView>(R.id.my_places_list)
    myPlacesList.adapter = ArrayAdapter<String>(view.context, android.R.layout.simple_list_item_1, myPlacesViewModel.myPlacesList)
}
```

- h. Probati aplikaciju ponovo!
- i. Preimenovati *FragmentFirst* u *HomeFragment* a *FragmentSecond* u *ListFragment*. Prilikom preimenovanja koristiti Refactor opciju dostupnu nakon desnog klika na fajl. Ovim će relevantne promene biti izvršene svuda u projektu. Ažurirati i relevantne xml fajlove u *fragment\_home.xml* i *fragment\_list.xml*, string resurse itd.



```

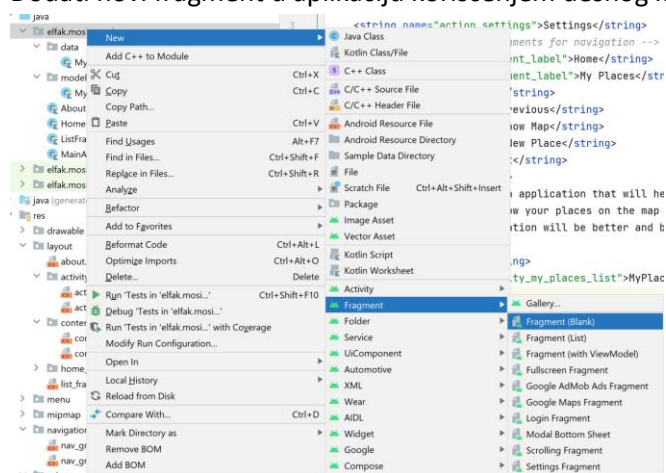
<resources>
    <string name="app_name">My Places</string>
    <string name="action_settings">Settings</string>
    <!-- Strings used for fragments for navigation -->
    <string name="first_fragment_label">Home</string>
    <string name="second_fragment_label">My Places</string>
    <string name="next">Next</string>
    <string name="previous">Previous</string>
    <string name="my_places_list">My Places List</string>
    <string name="show_map">Show Map</string>
    <string name="new_place">New Places</string>
    <string name="about">About</string>
    <string name="about_text">
        <!-- My Places --> is an application that will help you remember characteristics and locations of your favourite places.
        It will be able to show your places on the map and enable you to share them with your friends.
        Each week this application will be better and better and you will earn <i>points</i> from the teaching assistant
    </string>
    <string name="ok">OK</string>
</resources>

```

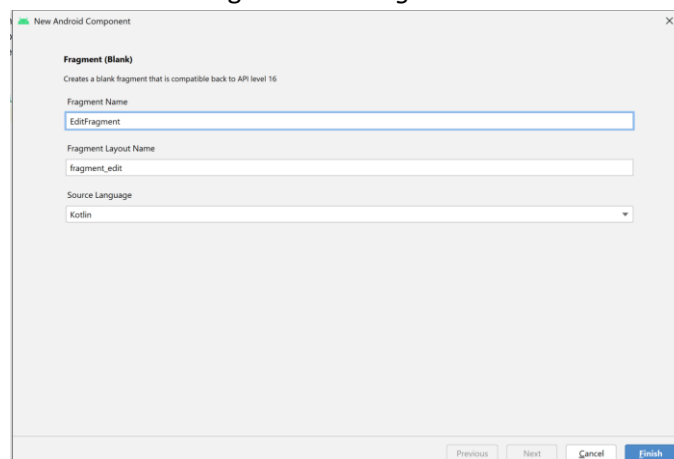
j. Probati aplikaciju i da li je sve u redu.

#### 4. Dodavanje fragmenta za unos nove omiljene lokacije.

a. Dodati novi fragment u aplikaciju korišćenjem desnog klika na paket projekta.

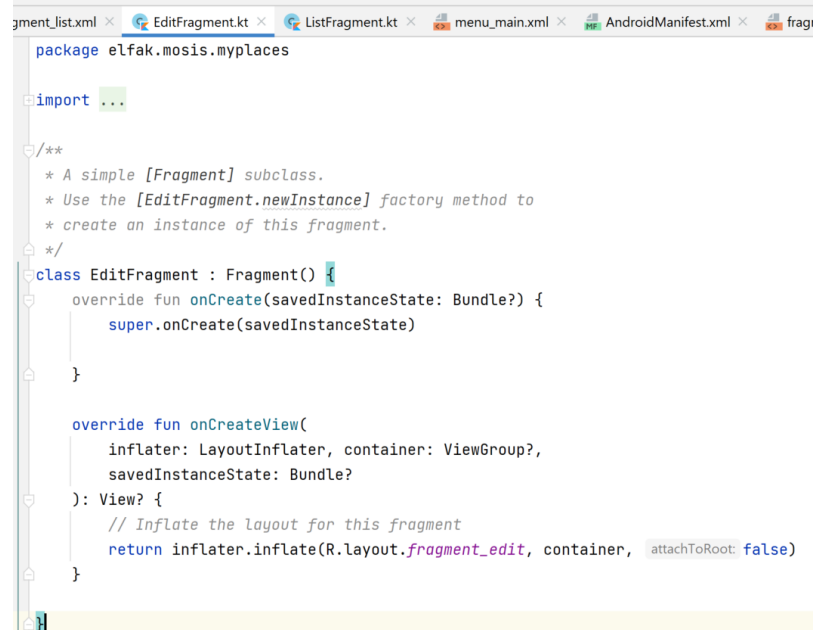


b. Imenovati novi fragment *EditFragment* a zatim kliknuti *Finish* dugme.



c. U novokreiranom fragmentu obrisati višak koda koji je anotiran sa TODO tako da klasa izgleda kao na slici.





```

package elfak.mosis.myplaces

import ...

/**
 * A simple [Fragment] subclass.
 * Use the [EditFragment.newInstance] factory method to
 * create an instance of this fragment.
 */
class EditFragment : Fragment() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
    }

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

```

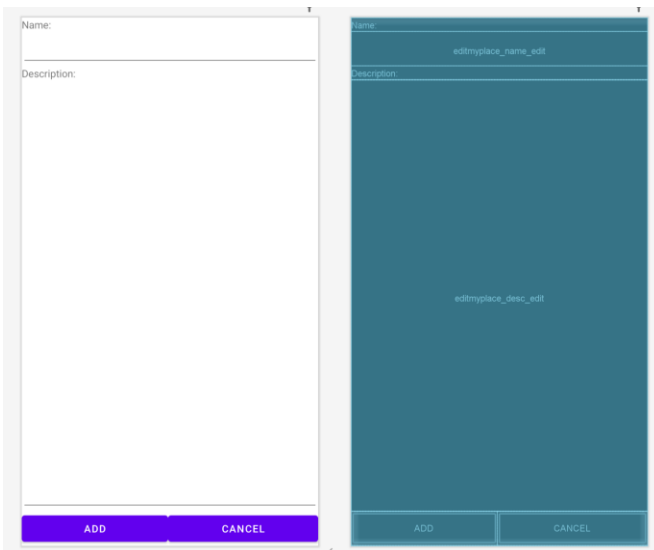
- d. U layout fajlu kreiranog *EditFragment*-a, *fragment\_edit.xml* prilagoditi korisnički interfejs da ima dve *TextView* kontrole i dve *EditText* kontrole. Dodati i dva dugmeta za dodavanje i otkazivanje dodavanja novog omiljenog mesta. U te svrhe moguće je koristiti *ConstraintLayout* ili *LinearLayout* kao u primeru.

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout 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=".EditFragment"
    android:orientation="vertical">
    <TextView
        android:id="@+id/textView2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/editmyplace_name_label" />
    <EditText
        android:id="@+id/editmyplace_name_edit"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:minHeight="48dp" />
    <TextView
        android:id="@+id/textView3"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/editmyplace_desc_label" />
    <EditText
        android:id="@+id/editmyplace_desc_edit"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1" />
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="bottom"
        android:orientation="horizontal">
        <Button
            android:id="@+id/editmyplace_finished_button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="@string/editmyplace_add_label" />
        <Button
            android:id="@+id/editmyplace_cancel_button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="@string/editmyplace_cancel_label" />
    </LinearLayout>
</LinearLayout>

```

- e. Nakon toga bi korisnički interfejs *EditFragment*-a trebalo da izgleda na sledeći način.

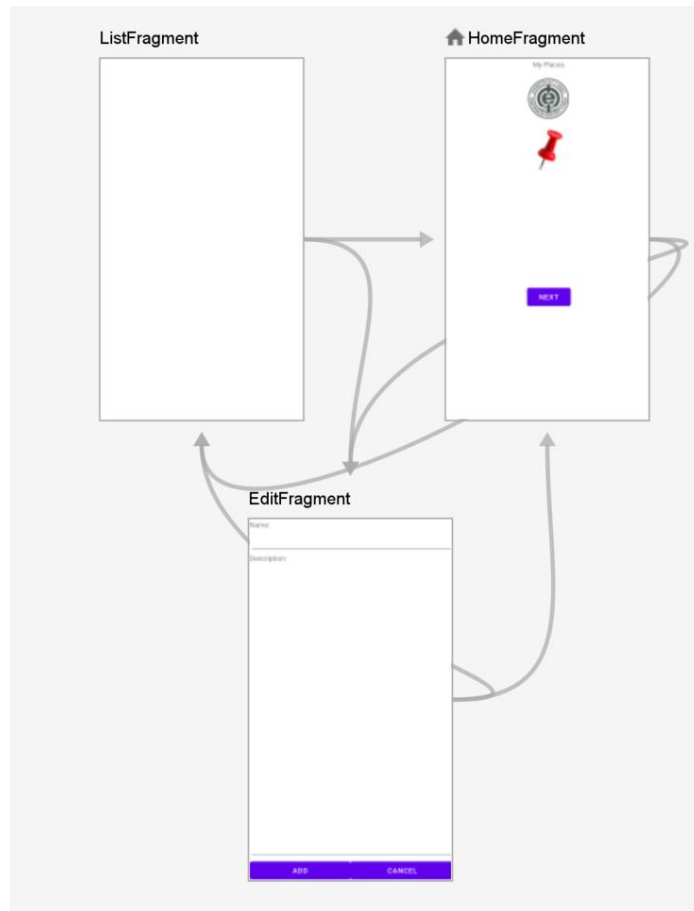


- f. Navigacija između fragmenata se vrši dodavanjem novog fragmenta u navigacioni graf. Potrebno je otvoriti *nav\_graph.xml* fajl i dodati novi fragment. Ažurirati i imena fragmenata prema novim imenima HomeFragment i ListFragment. Voditi računa o akcijama koje su definisane. Nakon dodavanja xml grafa treba da izgleda kao na slici.

```
<?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"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/nav_graph"
    app:startDestination="@id/HomeFragment">

    <fragment
        android:id="@+id/HomeFragment"
        android:name="elfak.mosis.myplaces.HomeFragment"
        android:label="@string/home_fragment_label"
        tools:layout="@layout/fragment_home">
        <action
            android:id="@+id/action_HomeFragment_to_ListFragment"
            app:destination="@id/ListFragment" />
        <action
            android:id="@+id/action_HomeFragment_to_EditFragment"
            app:destination="@id/EditFragment" />
    </fragment>
    <fragment
        android:id="@+id/ListFragment"
        android:name="elfak.mosis.myplaces.ListFragment"
        android:label="@string/list_fragment_label"
        tools:layout="@layout/fragment_list">
        <action
            android:id="@+id/action_ListFragment_to_HomeFragment"
            app:destination="@id/HomeFragment" />
        <action
            android:id="@+id/action_ListFragment_to_EditFragment"
            app:destination="@id/EditFragment" />
    </fragment>
    <fragment
        android:id="@+id/EditFragment"
        android:name="elfak.mosis.myplaces.EditFragment"
        android:label="@string/edit_fragment_label"
        tools:layout="@layout/fragment_edit">
        <action
            android:id="@+id/action_EditFragment_to_HomeFragment"
            app:destination="@id/HomeFragment" />
        <action
            android:id="@+id/action_EditFragment_to_ListFragment"
            app:destination="@id/ListFragment" />
    </fragment>
</navigation>
```

- g. Dok graf u dizajneru treba da izgleda kao na slici.



- h. Tranzicija između fragmenata će zavisiti od izbora elemenata na toolbar-u. Postoje dve opcije, kada opcioni meni kontroliše aktivnost koja sadrži fragmente ili kada ih kontrolišu sami fragmenti. U oba slučaja fragment može da utiče na rad menija. U ovom delu aplikacije biće korišćen meni opcija koji pripada aktivnosti. Potrebno je inicijalno omogućiti da fragment ima pristup meniju opcija. U sva tri fragmenta dodati sledeći kod.

```
override fun onCreate(savedInstanceState: Bundle?) {  
    super.onCreate(savedInstanceState)  
    setHasOptionsMenu(true)  
}  
  
override fun onCreateOptionsMenu(menu: Menu, inflater: MenuInflater) {  
    inflater.inflate(R.menu.menu_main, menu)  
}
```

- i. A zatim dodati i hendler za meni opcija za svaki od fragmenata
- j. *HomeFragment*:

```

override fun onOptionsItemSelected(item: MenuItem): Boolean {
    return when (item.itemId) {
        R.id.action_my_places_list -> {
            this.findNavController().navigate(R.id.action_HomeFragment_to_ListFragment)
            true
        }
        R.id.action_new_place -> {
            this.findNavController().navigate(R.id.action_HomeFragment_to_EditFragment)
            true
        }
        else -> super.onOptionsItemSelected(item)
    }
}

```

k. *ListFragment*:

```

override fun onOptionsItemSelected(item: MenuItem): Boolean {
    return when (item.itemId) {
        R.id.action_new_place -> {
            this.findNavController().navigate(R.id.action_ListFragment_to_EditFragment)
            true
        }
        else -> super.onOptionsItemSelected(item)
    }
}

```

l. *EditFragment*:

```

override fun onOptionsItemSelected(item: MenuItem): Boolean {
    return when (item.itemId) {
        R.id.action_my_places_list -> {
            this.findNavController().navigate(R.id.action_EditFragment_to_ListFragment)
            true
        }
        else -> super.onOptionsItemSelected(item)
    }
}

```

m. Meni mora da prikazuje samo one stavke koje su potrebne na svakom od fragmenata. Zbog toga je, na primer, potrebno onemogućiti klik na opciju za navigaciju na listu onda kada je lista već prikazana. Zbog toga je potrebno dodati i funkciju koja ažurira prikaz menija prema fragmentu koji je aktivan.

n. *ListFragment*:

```

override fun onPrepareOptionsMenu(menu: Menu){
    super.onPrepareOptionsMenu(menu)
    val item = menu.findItem(R.id.action_my_places_list)
    item.isVisible = false;
}

```

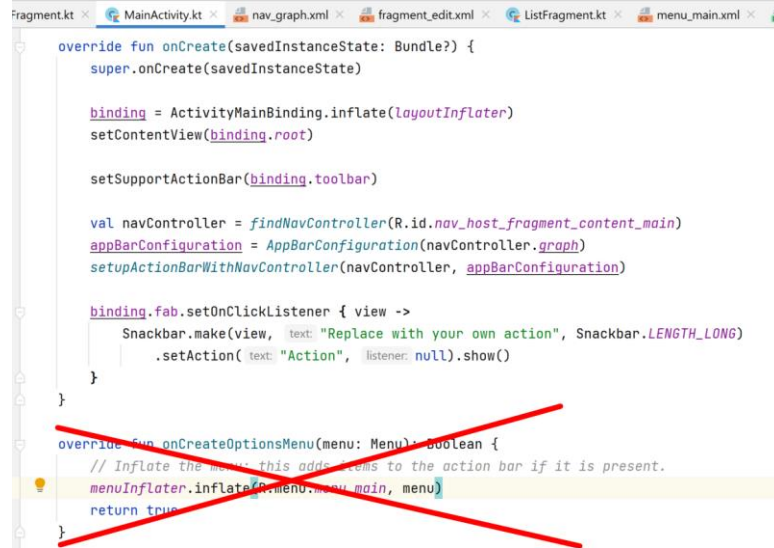
o. *EditFragment*:

```

override fun onPrepareOptionsMenu(menu: Menu){
    super.onPrepareOptionsMenu(menu)
    val item = menu.findItem(R.id.action_new_place)
    item.isVisible = false
}

```

- p. Potrebno je i ukloniti uvlačenje menija u okviru glavne aktivnosti.



```

Fragment.kt x MainActivity.kt x nav_graph.xml x fragment_edit.xml x ListFragment.kt x menu_main.xml x
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)

    binding = ActivityMainBinding.inflate(layoutInflater)
    setContentView(binding.root)

    setSupportActionBar(binding.toolbar)

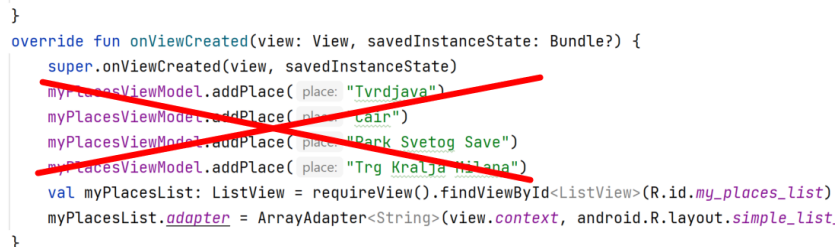
    val navController = findNavController(R.id.nav_host_fragment_content_main)
    appBarConfiguration = AppBarConfiguration(navController.graph)
    setupActionBarWithNavController(navController, appBarConfiguration)

    binding.fab.setOnClickListener { view ->
        Snackbar.make(view, text: "Replace with your own action", Snackbar.LENGTH_LONG)
            .setAction(text: "Action", listener: null).show()
    }
}

// Inflate the menu; this adds items to the action bar if it is present.
// onCreateOptionsMenu() method is crossed out with a red X
override fun onCreateOptionsMenu(menu: Menu): Boolean {
    menuInflater.inflate(R.menu.menu_main, menu)
    return true
}

```

- q. Probati aplikaciju i da li su meniji povezani.
- r. Obrisati dodavanje podataka u listu u okviru *ListFragment*-a.

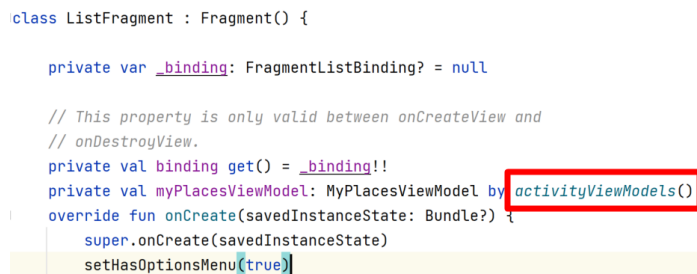


```

}
override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
    super.onViewCreated(view, savedInstanceState)
    myPlacesViewModel.addPlace(place: "Ivrdjava")
    myPlacesViewModel.addPlace(place: "Cair")
    myPlacesViewModel.addPlace(place: "Park Svetog Save")
    myPlacesViewModel.addPlace(place: "Trg Kralja Milana")
    val myPlacesList: ListView = requireView().findViewById<ListView>(R.id.my_places_list)
    myPlacesList.adapter = ArrayAdapter<String>(view.context, android.R.layout.simple_list_item_1, myPlacesViewModel.places)
}

```

- s. Potrebno je dodati logiku u *EditFragment* kojom se podaci preuzimaju iz EditText kontrola, kreira se novi *MyPlace* objekat i dodaje u *MyPlacesViewModel* instancu. Kako bi instance *ViewModel* klase mogla da se deli između svih fragmenata jedne aktivnosti, potrebno je izmeniti način kako se ona koristi okviru *ListFragment* klase a zatim istu deklaraciju dodati i u *EditFragment* klasu. Potrebno je da se umesto *viewModels()* delegata koristi *activityViewModels()*.



```

class ListFragment : Fragment() {

    private var _binding: FragmentListBinding? = null

    // This property is only valid between onCreateView and
    // onDestroyView.
    private val binding get() = _binding!!
    private val myPlacesViewModel: MyPlacesViewModel by activityViewModels()
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.fragment_list)
        setHasOptionsMenu(true)
    }
}

```

- t. Istu liniju koda dodati i u *EditFragment* a zatim dodati *click listener*-e na dugme *Add* i dugme *Cancel*.

```
override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
    super.onViewCreated(view, savedInstanceState)
    val addButton: Button = requireView().findViewById<Button>(R.id.editmyplace_finished_button)
    addButton.setOnClickListener { it: View!
        val editName: EditText = requireView().findViewById<EditText>(R.id.editmyplace_name_edit)
        val name: String = editName.text.toString()
        myPlacesViewModel.addPlace(name)
        findNavController().navigate(R.id.action_EditFragment_to_ListFragment)
    }
    val cancelButton: Button = requireView().findViewById<Button>(R.id.editmyplace_cancel_button)
    cancelButton.setOnClickListener { it: View!
        findNavController().navigate(R.id.action_EditFragment_to_ListFragment)
    }
}
```

- u. Probati aplikaciju.

- v. Ažurirati da *ViewModel* aplikacije radi sa listom *MyPlace* modela umesto sa stringovima.

```
MyPlacesViewModel.kt | MainActivity.kt | nav_graph.xml
package elfak.mosis.myplaces.model

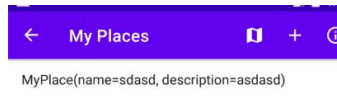
import androidx.lifecycle.ViewModel
import elfak.mosis.myplaces.data.MyPlace

class MyPlacesViewModel: ViewModel() {
    var myPlacesList: ArrayList<MyPlace> = ArrayList<MyPlace>()
    fun addPlace(place: MyPlace){
        myPlacesList.add(place);
    }
}

override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
    super.onViewCreated(view, savedInstanceState)
    val myPlacesList: ListView = requireView().findViewById<ListView>(R.id.my_places_list)
    myPlacesList.adapter = ArrayAdapter<MyPlace>(view.context, android.R.layout.simple_list_item_1, myPlacesViewModel.myPlacesList)
}

override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
    super.onViewCreated(view, savedInstanceState)
    val addButton: Button = requireView().findViewById<Button>(R.id.editmyplace_finished_button)
    addButton.setOnClickListener { it: View!
        val editName: EditText = requireView().findViewById<EditText>(R.id.editmyplace_name_edit)
        val name: String = editName.text.toString()
        val editDesc: EditText = requireView().findViewById<EditText>(R.id.editmyplace_desc_edit)
        val desc: String = editDesc.text.toString()
        myPlacesViewModel.addPlace(MyPlace(name, desc))
        findNavController().navigate(R.id.action_EditFragment_to_ListFragment)
    }
    val cancelButton: Button = requireView().findViewById<Button>(R.id.editmyplace_cancel_button)
    cancelButton.setOnClickListener { it: View!
        findNavController().navigate(R.id.action_EditFragment_to_ListFragment)
    }
}
```

- w. Probati aplikaciju. U listi se ispisuje generički izlaz *toString* funkcije *DataClass* objekta.



- x. Potrebno je izvršiti preklopiti funkciju *toString* u *DataClass*-i *MyPlace*.

```
MyPlaceFragment.kt x MyPlace.kt x MyPlacesViewModel.kt x ListFragment.kt x
package elfak.mosis.mylaces.data

data class MyPlace(var name:String, var description:String){
    override fun toString(): String = name
}
```

- y. Probati aplikaciju!

