Fragments

In software development and web design, the term "fragment" often refers to a small, reusable piece of code or user interface (UI). However, the specific meaning of "fragment" can vary based on the context. For instance, in Android development, a "Fragment" represents a behavior or a portion of the user interface in an Activity. Meanwhile, in web development, especially with frameworks like React, a "fragment" can denote a lightweight way to return multiple elements from a component.

In Android, a fragment represents a modular piece of an activity, which has its own lifecycle, receives its own input events, and can be added or removed while the activity is running.

Advantages of Using Fragments:

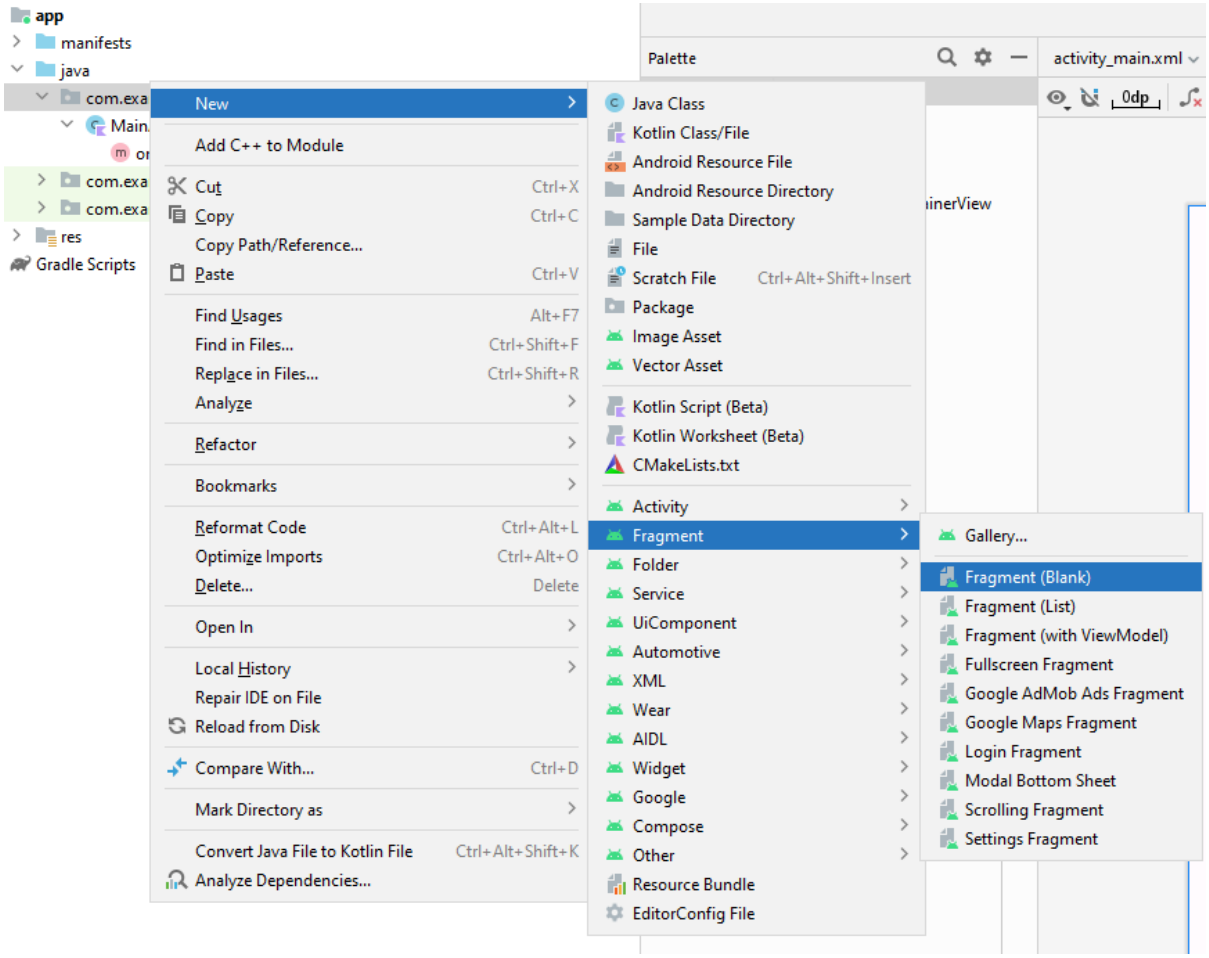
- **Reusability**: You can use the same fragment in multiple activities.
- **Adaptability**: With fragments, you can create more flexible and dynamic UIs that can adapt to different device configurations (e.g., tablets vs. phones).

Frame Layout

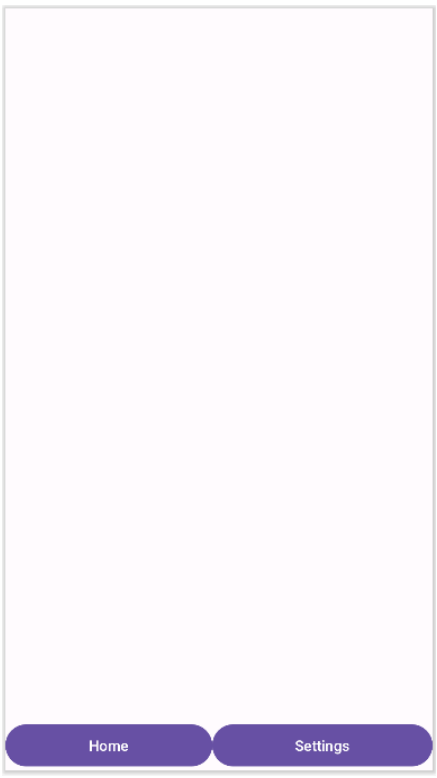
FrameLayout in Android is a simple layout manager designed to block out an area on the screen to display a single item, usually used to hold a single child view. If it holds multiple child views, they are stacked on top of each other with the last child view being at the top. FrameLayout allows specifying gravity to position its children, and its dimensions should generally match those of its largest child or be set to match_parent. It also supports a foreground drawable for overlay effects. For instance, in XML, a FrameLayout can be used to center a TextView over an ImageView, creating a visual overlay, making it a convenient choice for simple layouts and overlay effects.

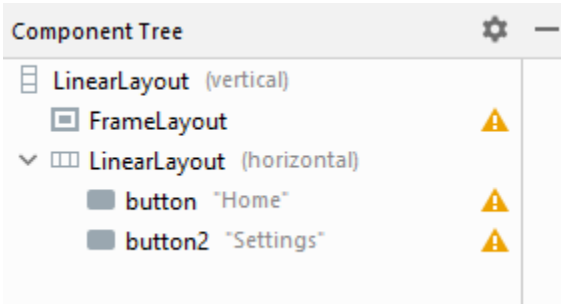
Example
Design

1. Create an empty views application.
2. Create a new Blank Fragment as follows.

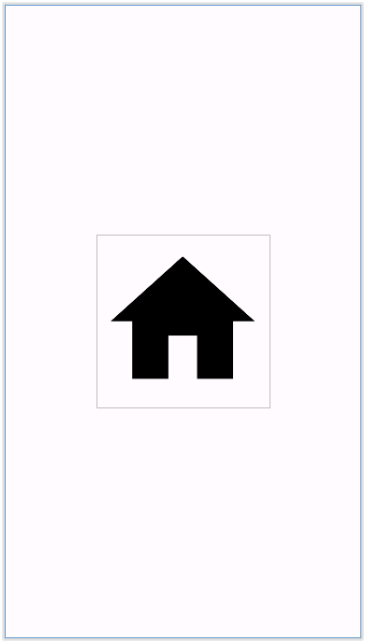


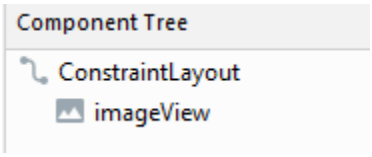
3. Name it as HomeFragement
4. Create another fragment named SettingsFragment.
5. Let's design the MainActivity as follows.



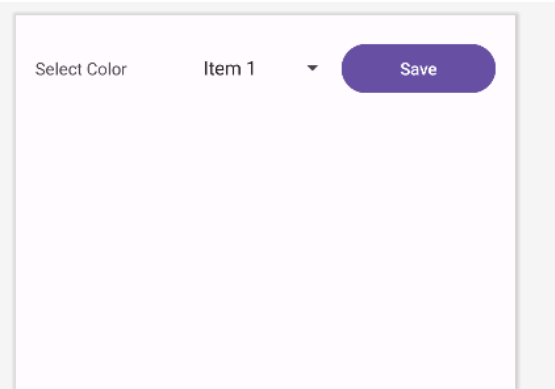


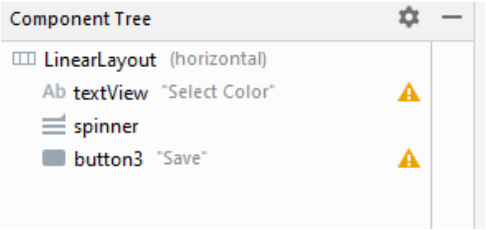
6. Now HomeFragment is as follows.





7. And the SettingsFragment is as follows.





Loading the Fragment

1. Create Fragment Objects in the MainActivity as follows.

```
val homeFragment = HomeFragment()
val settingsFragment = SettingsFragment()
```

2. Initialize the button views and implement the onClickListeners.

```
val button:Button = findViewById(R.id.button)
val button2:Button = findViewById(R.id.button2)

button.setOnClickListener {

}

button2.setOnClickListener {

}
```

3. Implement the loadHome() function

```
private fun loadHome(){
   val fragment = supportFragmentManager.findFragmentById(R.id.fragmentContainer)

   if (fragment == null){

supportFragmentManager.beginTransaction().add(R.id.fragmentContainer,homeFragment).commit()
   }else{

supportFragmentManager.beginTransaction().replace(R.id.fragmentContainer,homeFragment).comm
it()
   }
}
```

4. Implement the loadSettings() function

```
private fun loadSettings(){
      val fragment = supportFragmentManager.findFragmentById(R.id.fragmentContainer)

      if (fragment == null){

supportFragmentManager.beginTransaction().add(R.id.fragmentContainer,settingsFragment).commit
()
      }else{

supportFragmentManager.beginTransaction().replace(R.id.fragmentContainer,settingsFragment).com
mit()
      }
   }
```

5. Call each function from the relevant button click event.
6. Run the app and check the output.

Interacting with the Fragment

1. Update the onCreateView of the HomeFragment as follows:

```
override fun onCreateView(
   inflater: LayoutInflater, container: ViewGroup?,
   savedInstanceState: Bundle?
): View? {
   // Inflate the layout for this fragment
   val rootView = inflater.inflate(R.layout.fragment_home,container,false)

   val imageView:ImageView = rootView.findViewById(R.id.imageView)

   imageView.setOnClickListener {
      Toast.makeText(requireActivity(),"Home Button Clicked",Toast.LENGTH_LONG).show()
   }

   return rootView
}
```

2. Update the colors.xml with the following colors.

```
<color name="background1">#00B8D4</color>
<color name="background2">#FF6D00</color>
<color name="background3">#AEEA00</color>
<color name="background4">#00BFA5</color>
<color name="background5">#2962FF</color>
```

3. Update the strings.xml as follows:

```
<string-array name="colors">
   <item>Blank</item>
   <item>Background 1</item>
   <item>Background 2</item>
   <item>Background 3</item>
   <item>Background 4</item>
   <item>Background 5</item>
</string-array>
```

4. Set that string array to the entries of the spinner in the settings fragement.

5. Implement the onCreateView of the Settings fragment as follows:

```
override fun onCreateView(
    inflater: LayoutInflater, container: ViewGroup?,
    savedInstanceState: Bundle?
): View? {
    // Inflate the layout for this fragment
    val rootView = inflater.inflate(R.layout.fragment_settings,container,false)

    val spinner:Spinner = rootView.findViewById(R.id.spinner)
    val saveButton:Button = rootView.findViewById(R.id.button3)


        saveButton.setOnClickListener {

            when (spinner.selectedItem){
                "blank" -> {}
                "Background 1" -> {}
                "Background 2" -> {}
                "Background 3" -> {}
                "Background 4" -> {}
                "Background 5" -> {}

            }
        }
    return rootView
}
```

** In this app, the idea is to change the background color of the HomeFragment from the SettingsFragment. To do that we need to have a way to communicate with the two fragments.

Inter Fragment Communication

1. Create a class named FragmentViewModel
2. Extend the class from the ViewModel class.
3. Implement it as follows.

```
class FragmentViewModel: ViewModel() {

   private val backgroundColor = MutableLiveData<Int>()

   fun getBackgroundColor():LiveData<Int>{
      return backgroundColor
   }

   fun setBackground(color:Int){
      backgroundColor.value = color
   }
}
```

4. Modify the onCreateView() in the Settings Fragment as follows:

```
val viewModel = ViewModelProvider(requireActivity())[FragmentViewModel::class.java]

      saveButton.setOnClickListener {
         when (spinner.selectedItem){
            "blank" -> {}
            "Background 1" -> {viewModel.setBackground(R.color.background1)}
            "Background 2" -> {viewModel.setBackground(R.color.background2)}
            "Background 3" -> {viewModel.setBackground(R.color.background3)}
            "Background 4" -> {viewModel.setBackground(R.color.background4)}
            "Background 5" -> {viewModel.setBackground(R.color.background5)}

         }
      }
```

5. Now implement the observer in the HomeFragement as follows:

```
override fun onCreateView(
      inflater: LayoutInflater, container: ViewGroup?,
      savedInstanceState: Bundle?
   ): View? {
      // Inflate the layout for this fragment
      val rootView = inflater.inflate(R.layout.fragment_home,container,false)

      val imageView:ImageView = rootView.findViewById(R.id.imageView)
```

```
    imageView.setOnClickListener {
        Toast.makeText(requireActivity(),"Home Button Clicked",Toast.LENGTH_LONG).show()
    }

    val viewModel = ViewModelProvider(requireActivity())[FragmentViewModel::class.java]

    viewModel.backgroundColor.observe(requireActivity()){
        rootView.setBackgroundResource(it)
    }

    return rootView
}
```