**Section 34 – Android Jetpack Navigation Component (Java)**

**1. Key Concepts**

1. **Navigation Component Overview**
   * Part of Jetpack, helps implement **consistent, type-safe, and visual navigation** between screens.
   * Works with **fragments, activities, or custom destinations**.
   * Automatically:
     + Handles fragment transactions.
     + Manages **Up** and **Back** navigation correctly.
     + Supports **animations, transitions, and deep linking**.
     + Integrates with **navigation drawers** and **bottom navigation**.
   * Allows **safe argument passing** between destinations.
2. **Core Components**
   * **Navigation Graph (nav\_graph.xml)**
     + Centralized XML resource defining all destinations and possible navigation paths (actions).
   * **NavHostFragment**
     + A container that swaps destinations based on the navigation graph.
   * **NavController**
     + Java object that controls navigation within a NavHostFragment.
3. **Terminology**
   * **Destination** → Screen/UI element (Fragment, Activity, or custom).
   * **Action** → Path from one destination to another.
   * **Arguments** → Data passed between destinations.

**2. Step-by-Step Implementation**

**Step 1 – Add Dependency**

**build.gradle (app)**

dependencies {

def nav\_version = "2.7.4" // latest stable

implementation "androidx.navigation:navigation-fragment:$nav\_version" // For Java

implementation "androidx.navigation:navigation-ui:$nav\_version" // UI helpers

}

For Kotlin projects, use navigation-fragment-ktx and navigation-ui-ktx.

**Step 2 – Create Navigation Graph**

1. **Create a Navigation Resource Directory**
   * res > New > Android Resource Directory
   * Resource type: navigation.
2. **Add a Navigation Graph XML**
   * Name: navigation\_graph.xml.
   * Open in **Design View** for visual editing.

**Step 3 – Create Destinations (Fragments)**

**Example:**

1. Create **FirstFragment** (Blank Fragment, Java).
2. Create **SecondFragment**.

**Step 4 – Link Destinations with Actions**

* In **Design View**, click the **circle** on FirstFragment → drag to SecondFragment.
* This creates an <action> element in XML:

<action

android:id="@+id/action\_firstFragment\_to\_secondFragment"

app:destination="@id/secondFragment" />

* Repeat for back navigation:

<action

android:id="@+id/action\_secondFragment\_to\_firstFragment"

app:destination="@id/firstFragment" />

**Step 5 – Add NavHostFragment in Layout**

**activity\_main.xml**

<androidx.fragment.app.FragmentContainerView

android:id="@+id/nav\_host\_fragment"

android:name="androidx.navigation.fragment.NavHostFragment"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

app:defaultNavHost="true"

app:navGraph="@navigation/navigation\_graph" />

* app:defaultNavHost="true" → Makes it handle system Back button automatically.
* app:navGraph → Links layout to navigation graph.

**Step 6 – Add UI & Handle Navigation**

**fragment\_first.xml**

<Button

android:id="@+id/btn\_one"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Go to Second" />

**FirstFragment.java**

public class FirstFragment extends Fragment {

@Nullable

@Override

public View onCreateView(@NonNull LayoutInflater inflater,

@Nullable ViewGroup container,

@Nullable Bundle savedInstanceState) {

View view = inflater.inflate(R.layout.fragment\_first, container, false);

Button button = view.findViewById(R.id.btn\_one);

button.setOnClickListener(v ->

Navigation.findNavController(v)

.navigate(R.id.action\_firstFragment\_to\_secondFragment)

);

return view;

}

}

**Key Points:**

* Navigation.findNavController(view) → Gets NavController for the current NavHostFragment.
* .navigate(actionId) → Executes navigation action.

**Step 7 – Back Navigation**

**SecondFragment.java**

public class SecondFragment extends Fragment {

@Nullable

@Override

public View onCreateView(@NonNull LayoutInflater inflater,

@Nullable ViewGroup container,

@Nullable Bundle savedInstanceState) {

View view = inflater.inflate(R.layout.fragment\_second, container, false);

Button button = view.findViewById(R.id.btn\_two);

button.setOnClickListener(v ->

Navigation.findNavController(v)

.navigate(R.id.action\_secondFragment\_to\_firstFragment)

);

return view;

}

}

**Step 8 – Test**

1. Run app.
2. Click "Go to Second" → Should navigate to second fragment.
3. Click "Back to First" → Returns to first fragment.

**3. Tools, Libraries, APIs Used**

* **AndroidX Navigation Component**
  + navigation-fragment (Fragment-based navigation)
  + navigation-ui (UI helpers for menus, toolbars, drawers)
* **FragmentContainerView** – Recommended container for fragments.
* **Navigation.findNavController()** – Primary API for programmatic navigation.

**4. Best Practices & Industry Tips**

* Use **Safe Args** plugin for type-safe argument passing.
* Prefer **single-activity architecture** with multiple fragments.
* Keep navigation logic inside **UI controllers** (fragments, not ViewModels).
* Keep fragment transactions **out of business logic**; always use NavController.
* Always give clear, consistent IDs to actions and destinations.
* For global navigation (e.g., logout), define **global actions** in the nav graph.
* Use **NavigationUI.setupActionBarWithNavController()** to integrate with Toolbar.

**5. Part B – Extra Knowledge Not Covered**

1. **Safe Args (Strongly Typed Navigation)**
   * Add in build.gradle (project):
   * classpath "androidx.navigation:navigation-safe-args-gradle-plugin:2.7.4"
   * In build.gradle (app):
   * apply plugin: "androidx.navigation.safeargs"
   * Generates classes for type-safe argument passing.
2. **Deep Linking**
   * Allows navigation directly to a destination from outside the app (e.g., URL click).
3. **Pop Behavior**
   * Control backstack with:
   * navController.popBackStack();
   * Or:
   * app:popUpTo="@id/homeFragment"
   * app:popUpToInclusive="true"
4. **Navigation Animations**
   * Add enterAnim, exitAnim attributes to actions in nav graph.
5. **Navigation with BottomNavigationView**
   * Use NavigationUI.setupWithNavController(bottomNav, navController).
6. **Multiple NavHostFragments**
   * Useful for multi-pane or tabbed UIs.