<https://developer.android.com/training/implementing-navigation/nav-drawer.html>

<http://blog.teamtreehouse.com/add-navigation-drawer-android>

selectItem:

1. Situations:
   1. **Go into unitFragment when both unitFragment and daysUntilFragment have never been touched before**
      1. 🡺 create a new UnitFragment
      2. add
   2. (Currently at unitFragment) Go into daysUntilFragment when unitFragment is currently showing
      1. Replace()
   3. Go into unitFragment when unitFragment is currently showing
      1. Don’t do anything
   4. **Go into daysUntilFragment when both unitFragment and daysUntilFragment have never been touched before**
      1. 🡺 create a new DaysUntilFragment
      2. add
   5. (Currently at daysUntilFragment) Go into unitFragment when daysUntilFragment is currently showing
   6. Go into daysUntilFragment when daysUntilFragment is currently showing
      1. Don’t do anything
2. Methods to look over
   1. FragmentTransaction:
      1. Add
      2. Replace
         1. Does 2 things:
      3. Remove
      4. addTobackStack(String name)
         1. <https://developer.android.com/reference/android/app/FragmentTransaction.html#addToBackStack(java.lang.String)>
         2. Add this transaction to the back stack. This means that the transaction will be remembered after it is committed, and will reverse its operation when later popped off the stack.
   2. FragmentManager
      1. popFromBackStack
         1. reverses your last transaction that you've added to BackStack.
   3. <http://stackoverflow.com/questions/17793249/how-do-popbackstack-and-replace-operations-differ>

|  |  |
| --- | --- |
| 48 down vote accepted | replace() does 2 things:   1. Remove currently added fragment (A) from the container (C) you indicated 2. Add new fragment (B) to the same container   These 2 operations are what is saved as a Backstack record / transaction. Note that fragment A remains in created state, and its view is destroyed.  Now popBackStack() reverses your last transaction that you've added to BackStack.  In this case that would be 2 steps:   1. Remove B from C 2. Add A to C   After this, fragment B becomes detached, and if you did not keep references to it, it will be garbage collected.  To answer first part of your question, there's no onCreate() call, because FragmentB remained in created state. And answer to second part of the question is a bit longer.  First, it is important to understand that you don't actually add Fragments to Backstack, you add FragmentTransactions. So when you think that you "replace with Fragment B, adding Fragment A to the back stack", you actually add this whole operation to backstack - that is *replacement* of A with B. This replacement consists of 2 actions - remove A and add B.  Then, next step is popping of the transaction that contains this replacement. So you're not popping FragmentA, you're reversing "remove A, add B", which reversed is "remove B, add A".  And then final step should be clearer - there's no B that FragmentManager is aware of, so when you add it by replacing A with B at your last step, B needs to go through its early lifecycle methods - onAttach() and onCreate(). |