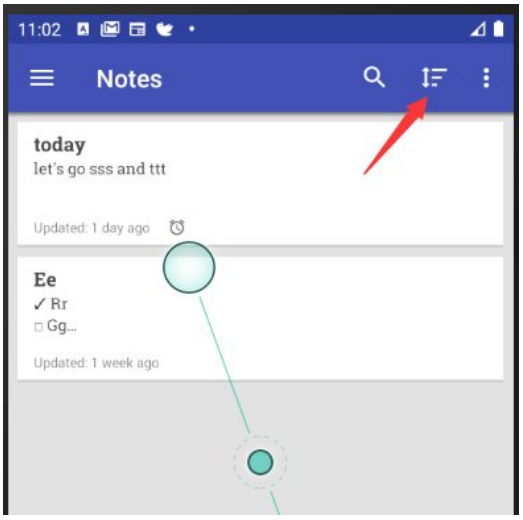
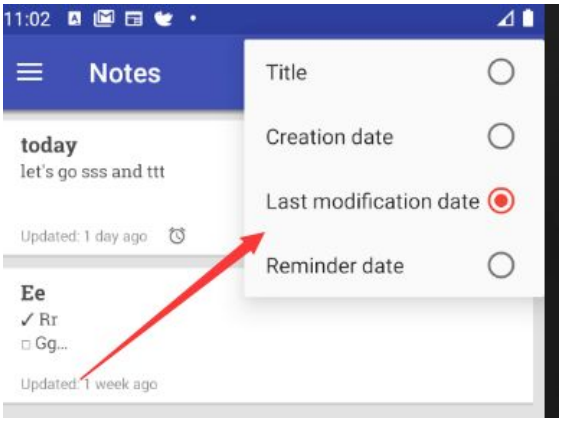
Sorting

Basic modules of sorting are in the *ListFragment.java*. Because from the screen, we can see that the button of sorting submenu is in the Note-List interface, as shown in the screenshot below:



and the submenu is like this:



We can sort the existing notes by title, creation date, last modification date and reminder date.

So we searched through the *ListFragment.java* and we found **initSortingSubmenu ()**. Just as its name, it's used to initiate this submenu. And in this module, it gives all items in this submenu a group_id: ***MENU_SORT_GROUP_ID,*** and set the first item as selected. And it also makes sure to set the parameter, ***PREF_SORTING_COLUMN***, which is saved in *sharedpreference*, as the first item in this submenu. **initSortingSubmenu ()** is shown below:

```
private void initSortingSubmenu () {
  final String[] arrayDb = getResources().getStringArray(R.array.sortable_columns);
  final String[] arrayDialog = getResources().getStringArray(R.array.sortable_columns_human_readable);
  int selected = Arrays.asList(arrayDb).indexOf(prefs.getString(PREF_SORTING_COLUMN, arrayDb[0]));

  SubMenu sortMenu = this.menu.findItem(R.id.menu_sort).getSubMenu();
  for (int i = 0; i < arrayDialog.length; i++) {
    if (sortMenu.findItem(i) == null) {
      sortMenu.add(MENU_SORT_GROUP_ID, i, i, arrayDialog[i]);
    }
    if (i == selected) {
      sortMenu.getItem(i).setChecked(true);
    }
  }
  sortMenu.setGroupCheckable(MENU_SORT_GROUP_ID, b: true, b1: true);
}
```

Then we use “find usage” to see where we call this module, and we found **performAction()**, and according to the comment we can know that this function is called when an ActionBar’s Button is pressed:

```
/**
 * Performs one of the ActionBar button's actions after checked notes protection
 */
public void performAction (MenuItem item, ActionMode actionMode) {
```

and we can get this:

```
case R.id.menu_sort:
  initSortingSubmenu();
  break;
```

And at the end of **performAction()**, we found another module named **checkSortActionPerformed()** as follows:

```
private void checkSortActionPerformed (MenuItem item) {
  if (item.getGroupId() == MENU_SORT_GROUP_ID) {
    final String[] arrayDb = getResources().getStringArray(R.array.sortable_columns);
    prefs.edit().putString(PREF_SORTING_COLUMN, arrayDb[item.getOrder()]).apply();
    initNotesList(mainActivity.getIntent());
    // Resets list scrolling position
    listViewPositionOffset = 16;
    listViewPosition = 0;
    restoreListScrollPosition();
    toggleSearchLabel( activate: false);
    // Updates app widgets
    mainActivity.updateWidgets();
  } else {
    ((OmniNotes) getActivity().getApplication()).getAnalyticsHelper().trackActionFromResourceId(getActivity(),
        item.getItemId());
  }
}
```

As I said above, we gave the items in sorting submenu a group_id, and we need that group_id to do further operations, so we can make sure that in this module is the real place to sort notes, we can also ensure this fact from the module’s name.

We can see that in this module, we set the ***PREF_SORTING_COLUMN*** as the name of the item we selected and then initiate the whole list. Because we use that name as the sorting order, we are going to see what happened in the **initNoteList()**.

```
void initNotesList (Intent intent) {
    LogDelegate.d("initNotesList intent: " + intent.getAction());

    progress_wheel.setAlpha(1);
    list.setAlpha(0);

    //...
    if (Intent.ACTION_VIEW.equals(intent.getAction()) && intent.getCategories() != null
            && intent.getCategories().contains(Intent.CATEGORY_BROWSABLE)) {...}

    if (ACTION_SHORTCUT_WIDGET.equals(intent.getAction())) {...}

    // Searching
    searchQuery = searchQueryInstant;
    searchQueryInstant = null;
    if (searchTags != null || searchQuery != null || searchUncompleteChecklists
            || IntentChecker.checkAction(intent, Intent.ACTION_SEARCH, ACTION_SEARCH_UNCOMPLETE_CHECKLISTS)) {...} else {
        // Check if is launched from a widget with categories
        if ((ACTION_WIDGET_SHOW_LIST.equals(intent.getAction()) && intent.hasExtra(INTENT_WIDGET))
                || !TextUtils.isEmpty(mainActivity.navigationTmp)) {...} else {
            NoteLoaderTask.getInstance().execute0nExecutor(AsyncTask.THREAD_POOL_EXECUTOR, ...params: "getAllNotes", true);
        }
    }
}
```

initNoteList() is a big module and most of the contents are about the searching function. But we can still find that at the end of the module, the **NoteLoaderTask** executed a function called **getAllNotes**.

So we found the **getAllNotes()** in *DBHelper.java* and the final step of reference is the **getNotes()**, some part of this module is showed below:

```
sortColumn = prefs.getString(PREF_SORTING_COLUMN, KEY_TITLE);
```

```
// Generic query to be specialized with conditions passed as parameter
String query = "SELECT "
    + KEY_CREATION + ","
    + KEY_LAST_MODIFICATION + ","
    + KEY_TITLE + ","
    + KEY_CONTENT + ","
    + KEY_ARCHIVED + ","
    + KEY_TRASHED + ","
    + KEY_REMINDER + ","
    + KEY_REMINDER_FIRED + ","
    + KEY_RECURRENCE_RULE + ","
    + KEY_LATITUDE + ","
    + KEY_LONGITUDE + ","
    + KEY_ADDRESS + ","
    + KEY_LOCKED + ","
    + KEY_CHECKLIST + ","
    + KEY_CATEGORY + ","
    + KEY_CATEGORY_NAME + ","
    + KEY_CATEGORY_DESCRIPTION + ","
    + KEY_CATEGORY_COLOR
    + " FROM " + TABLE_NOTES
    + " LEFT JOIN " + TABLE_CATEGORY + " USING( " + KEY_CATEGORY + ") "
    + whereCondition
    + (order ? " ORDER BY " + sortColumn + " COLLATE NOCASE " + sortOrder : "");
```

Finally we found that the ***PREF_SORTING_COLUMN*** saved in *sharereference* is used here, as the *order by* parameter in the query sentence. So with the query built, we can get a result set

with the order we selected in sorting submenu and rebuild the whole list, and replace the old list with this new one. That’s how the sorting function works.

Call Graph for This Feature