

Mobile Computing

Practice # 2b

Android Applications – Interface (in Kotlin)

Another step in the restaurants application

1. Adding a menu item and notes to the restaurant application

Let's allow the user to add a few lines of notes about each restaurant and to see those notes with a pop-up message (known as a toast) when in the **List** tab.

- Start by adding a 'notes' string property to the Restaurant class.
- Add a new **EditText** view to the interface with a maximum of two lines, like the example below.

The screenshot shows a mobile application interface for 'Lunch List'. At the top, there's a teal header with a chef's hat icon and the text 'Lunch List'. Below the header, there are two tabs: 'LIST' and 'DETAILS'. The 'DETAILS' tab is selected, indicated by a pink underline. The form contains the following fields: 'Name:' with the value 'Foo Bar', 'Address:' with the value '1313 Mockingbird Lane', 'Type:' with three radio button options: 'Take-Out', 'Sit-Down' (which is selected with a pink dot), and 'Delivery'. Below these is a 'Notes:' field with the text 'The food is excellent but the wait staff is a bit strange ...'. At the bottom of the form is a grey 'SAVE' button. The status bar at the top shows the time '16:46' and various icons. The bottom of the screen shows the standard Android navigation bar with back, home, and recent apps buttons.

- Define a main menu with a single item with title "Raise Toast" and a suitable icon. The icon should be a 32x32 .png file residing in the drawable folder (like the supplied *toast.png*). The menu definition should be a XML file on the menu folder (like *main.xml*). The menu item option `android:showAsAction="ifRoom"` will make this item and its icon appear in the **ActionBar** instead of in the general menu icon (a three dots icon).
- Arrange for the menu to be displayed overriding the `onCreateOptionsMenu()` method of the Activity:

```
override onCreateOptionsMenu(menu: Menu?): Boolean {  
    MenuInflater(this).inflate(R.menu.main, menu)  
    return super.onCreateOptionsMenu(menu)  
}
```

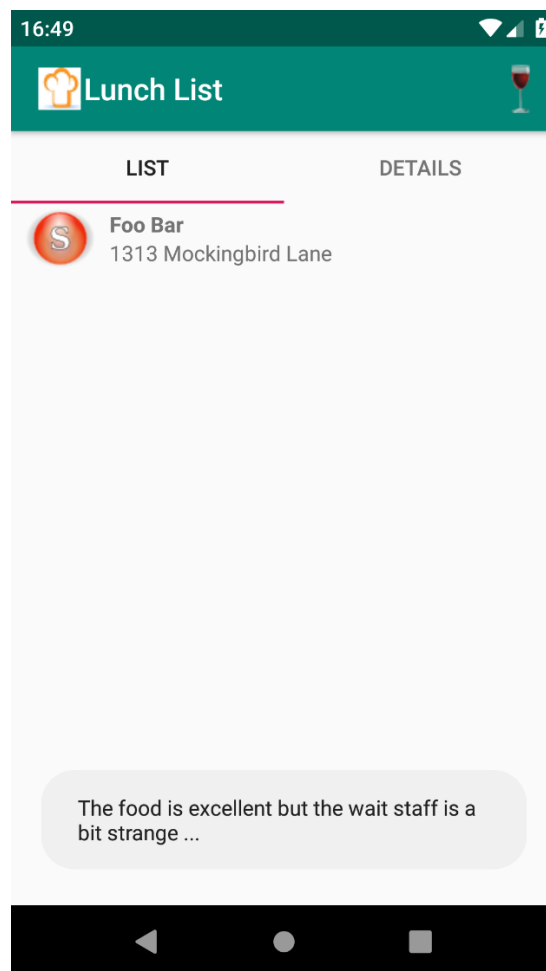
- e. Finally show the **notes** as a toast window when the user selects the menu item. To do this, we need the notion of a **current** restaurant and so we need a data member property of the Activity class (for example named **current**) as a Restaurant and initialized to **null**.

Whenever the user saves a new restaurant (in the handler of the **save button**) or selects one on the list (in the **onItemClick** handler) the current data member should be filled with all the details.

For showing the toast window our activity should have an overriding of the **onOptionsItemSelected()** method, like the following code:

```
override onOptionsItemSelected(item: MenuItem): Boolean {  
    if (item.getItemId() == R.id.toast) {  
        val message = current?.notes ?: "No restaurant selected"  
        Toast.makeText(this, message, Toast.LENGTH_LONG).show()  
        return true  
    }  
    return super.onOptionsItemSelected(item)  
}
```

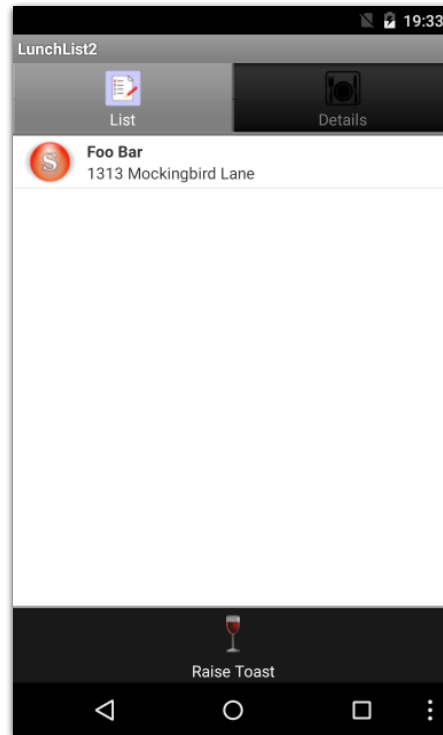
The result should be like the next screen:



- f. If you use a style previous to Android version 3.0 (like **Theme.Black** or **Theme.Light**) maintaining the target as **API 11** or after, the **ActionBar** is replaced by a **TitleBar** and, if the device does not have a hard menu button, it becomes inaccessible. You will have a menu in the zone of the soft buttons, even if you don't have a menu hard button, only if you change the min and target SDK version to anything before **Honeycomb (API 11)**, for instance, **API 10** (associated with Android 2.3.3). In this case you will have an old style menu like the one shown in next the picture.

Optionally try other implementations:

1. Try to use an **AlertDialog** instead of a **Toast**.
2. Try to use a **SnackBar** (in one of the Android support libraries) instead of a **Toast** (notice the increase in the .apk file size).
3. Try to use a menu button to switch between tabs. In particular, change the text and icon on the menu option to reflect the other tab (i.e., on the List tab, the menu should show "Details" and the details tab icon; on the Details tab, the menu should show "List" and the List tab icon).
4. Try using an **AlertDialog** designed to display exceptions in a "pleasant" format to the end user. The **AlertDialog** should also log the exceptions via **android.util.Log**. Use some sort of runtime exception (e.g., division by zero) for generating exceptions to pass to the dialog.



An old style menu (just one item) and tabs
(can be seen this way in newer devices if the target API is less than 11)