Mobile Computing

Practice # 2b

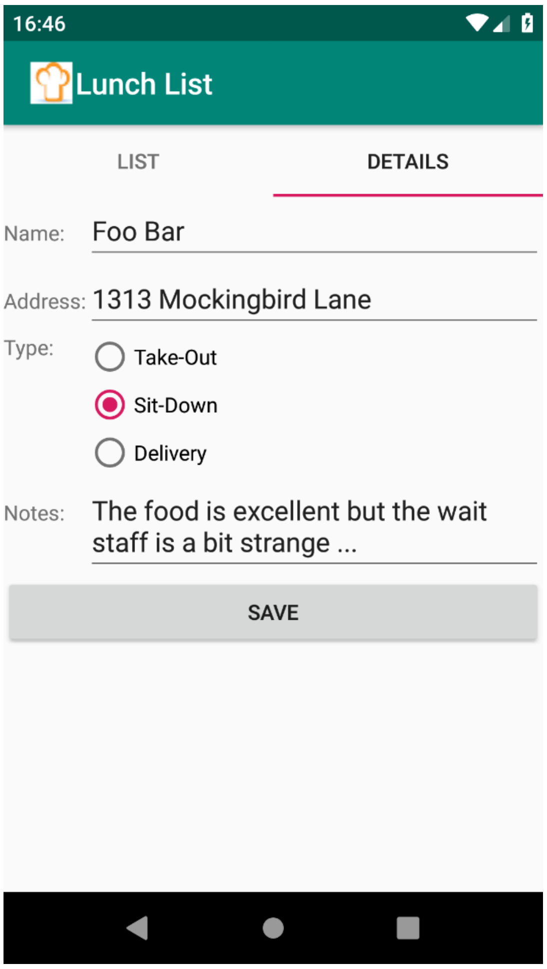
Android Applications - Interface

Another step in the restaurant's app

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

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

a. Start by adding a 'notes' string field to the Restaurant class and its getter and setter.
b. Add a new EditText view to the interface with a maximum of two lines, like the example below.



Name: Foo Bar

Address: 1313 Mockingbird Lane

Type: Take-Out [✓ Sit-Down] Delivery

Notes: The food is excellent but the wait staff is a bit strange ...

c. 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 an 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 (represented by **three dots**).
d. Arrange for the menu to be displayed overriding the onCreateOptionsMenu() method in the Activity as:

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    new MenuInflater(this).inflate(R.menu.main, menu);
    return (super.onCreateOptionsMenu(menu));
}
```

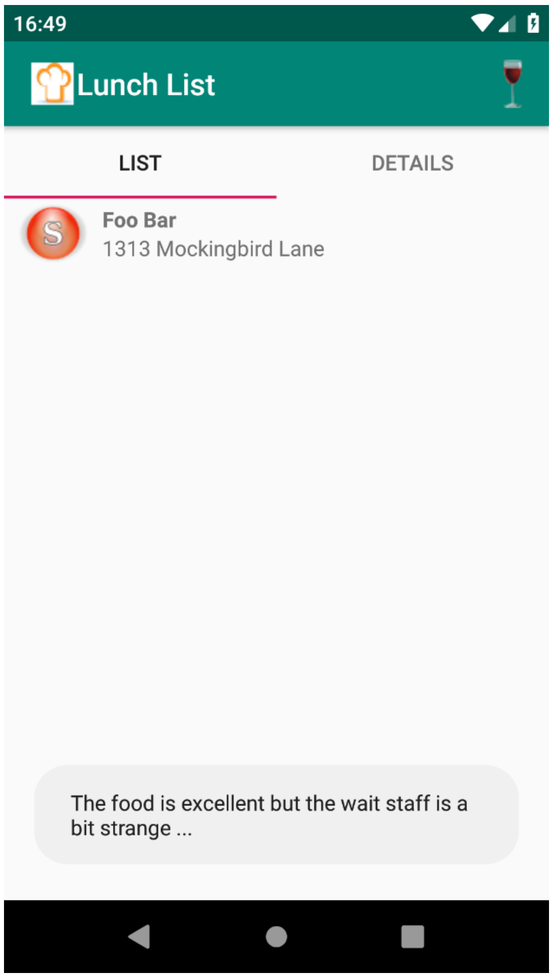
e. Finally, show the notes as a toast window when the user selects the menu item. To do this, we need the notion of a current restaurant and so we need a data member variable of the Activity class (for example named current) as a Restaurant and initialized to null.

   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 (for the main menu), like the following code:

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    if (item.getItemId()==R.id.toast) {
        String message="No restaurant selected";
        if (current!=null)
            message=current.getNotes();
        Toast.makeText(this, message, Toast.LENGTH_LONG).show();
        return(true);
    }
    return(super.onOptionsItemSelected(item));
}
```

   The result should be like the next screen:



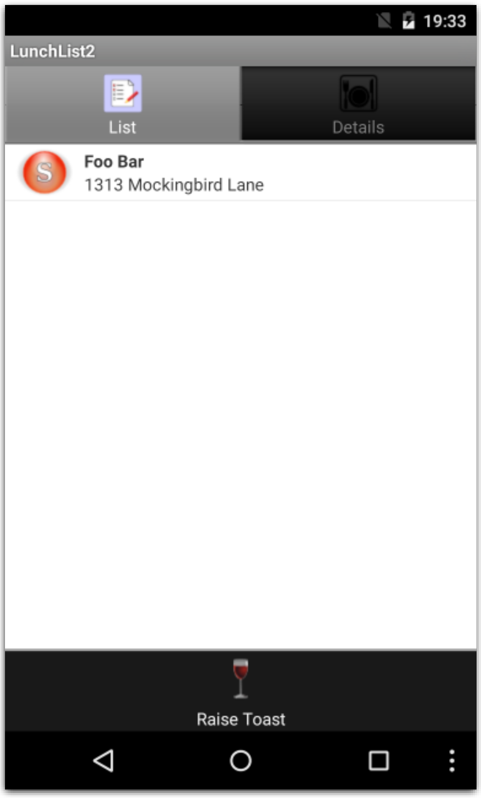
The food is excellent but the wait staff is a bit strange ...

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 the next figure.

Optionally try other implementations:

1. Try to use an **AlertBox** (it is a kind of pre-defined dialog box) 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 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 **ErrorDialog** designed to display exceptions in a "pleasant" format to the end user. The **ErrorDialog** 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 10)