Menus & Preferences

Essential Android Skills

Ref: Chapter 8 Murach's Android Programming 2Ed

Learning Outcomes

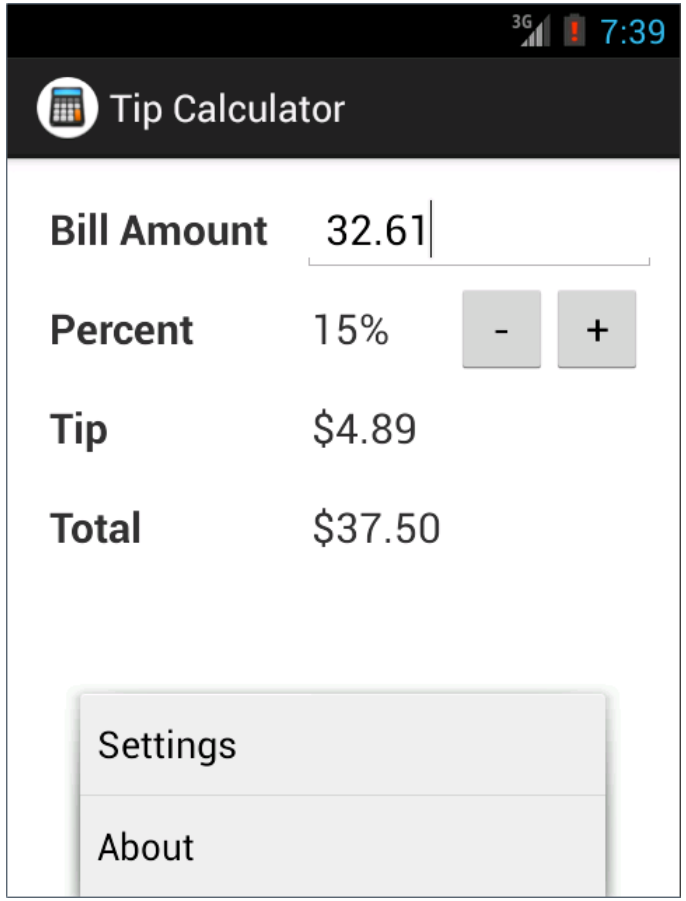
Applied

- Given the specifications for a menu that contains one or more items, define the menu and its items and handle the events that occur when users select those items.
- Given the specifications for the preferences for an app, use a fragment to allow the user to set those preferences.

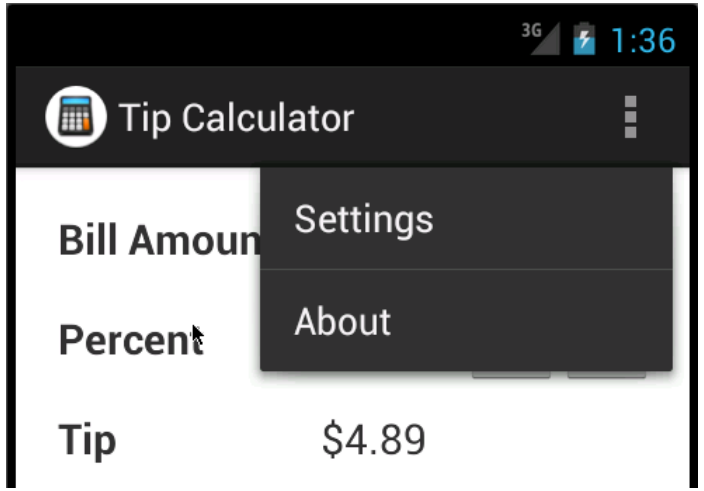
Knowledge

- Name and describe the most common type of menu.
- Distinguish between an action item and the action overflow item.
- Describe how Android decides whether to display an item as an action item or a menu item.
- Explain how to use an intent to start an activity.
- In general, explain how the Preferences API works and list one of its benefits.

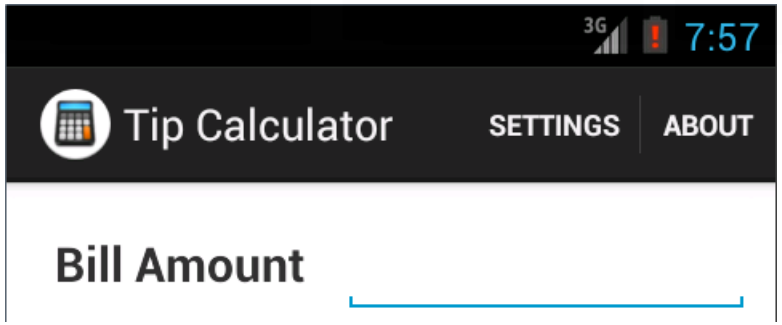
An activity with an options menu that has two items



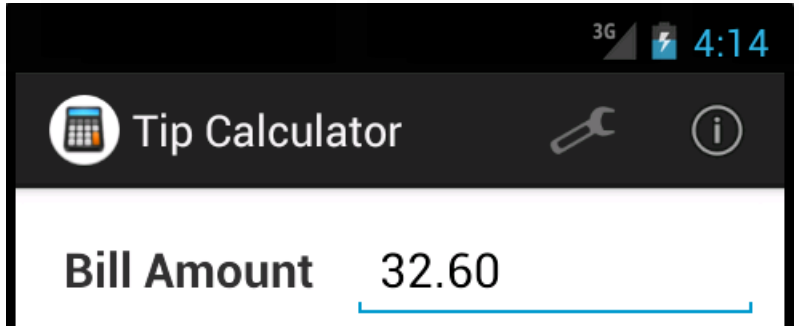
The same options menu displayed from an action overflow icon



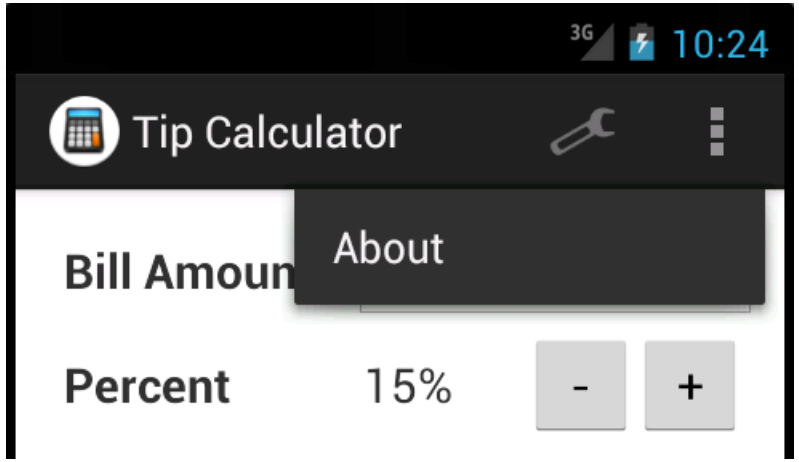
Two action items displayed as text



Two action items displayed as icons



One action icon and an overflow action icon



Three types of menus

- An *options menu* typically drops down from the action overflow icon as shown earlier. However, it can also be displayed across the bottom of the screen as shown earlier.
- A *floating context menu* appears as a floating list of menu items when a user performs a long click on the widget. Alternately, *contextual action mode* can display action items that apply to the selected item or items.
- A *popup menu* usually appears as a list below a widget or action item.

The file that contains the XML for the menu

```
res\menu\activity_tip_calculator.xml
```

The XML for the menu

```
<menu xmlns:android=
        "http://schemas.android.com/apk/res/android">
    <item android:id="@+id/menu_settings"
          android:title="@string/menu_settings"
          android:icon="@drawable/ic_settings"
          android:showAsAction="ifRoom" />
    <item android:id="@+id/menu_about"
          android:title="@string/menu_about"
          android:icon="@drawable/ic_about"
          android:showAsAction="never" />
</menu>
```

Some attributes of a menu item

```
title
icon
showAsAction
orderInCategory
```

Typical values for the showAsAction attribute

```
always
never
ifRoom
```

The location of the icon files for the items

```
res\drawable-xhdpi\ic_settings.png
res\drawable-xhdpi\ic_about.png
```

A directory that has standard icons for API 23

```
\sdk\platforms\android-23\data\res\drawable-xhdpi
```

The code that displays the menu

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(
        R.menu.activity_tip_calculator, menu);
    return true;
}
```

The code that handles the menu item events

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.menu_settings:
            Toast.makeText(this, "Settings",
                Toast.LENGTH_SHORT).show();
            return true;
        case R.id.menu_about:
            Toast.makeText(this, "About",
                Toast.LENGTH_SHORT).show();
            return true;
        default:
            return super.onOptionsItemSelected(item);
    }
}
```

Code that starts a new activity

Two statements

```
Intent settings = new Intent(
    getApplicationContext(), SettingsActivity.class);
startActivity(settings);
```

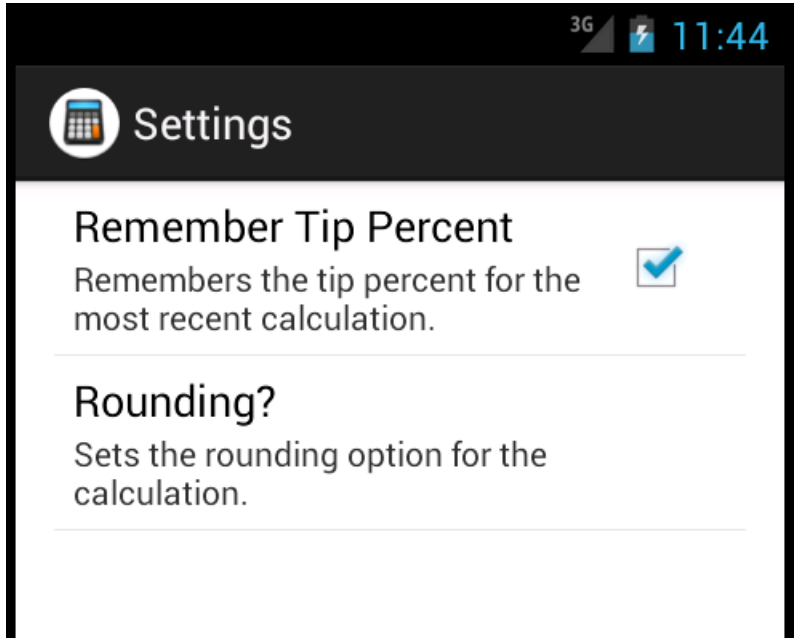
One statement

```
startActivity(new Intent(
    getApplicationContext(), SettingsActivity.class));
```

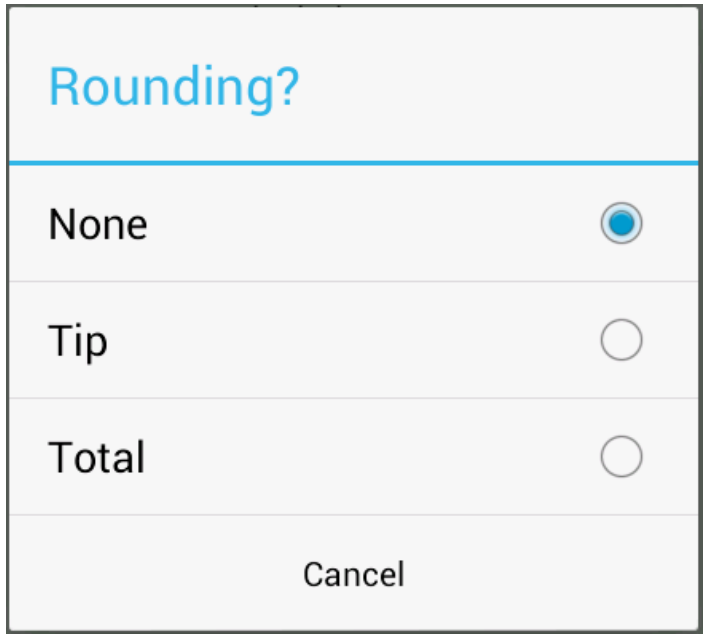
Code that uses menu items to start new activities

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.menu_settings:
            startActivity(new Intent(
                getApplicationContext(),
                    SettingsActivity.class));
            return true;
        case R.id.menu_about:
            startActivity(new Intent(
                getApplicationContext(),
                    AboutActivity.class));
            return true;
        default:
            return super.onOptionsItemSelected(item);
    }
}
```

The Settings activity



The dialog for the “Rounding?” item



The file that contains the XML for the preferences

res\xml\preferences.xml

The XML for the preferences

```
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen xmlns:android=
        "http://schemas.android.com/apk/res/android">
    <CheckBoxPreference
        android:key="pref_remember_percent"
        android:title="@string/remember_percent_title"
        android:summary="@string/remember_percent_summary"
        android:defaultValue="true" />
    <ListPreference
        android:key="pref_rounding"
        android:title="@string/rounding_title"
        android:summary="@string/rounding_summary"
        android:dialogTitle="@string/rounding_title"
        android:entries="@array/rounding_keys"
        android:entryValues="@array/rounding_values"
        android:defaultValue="@string/rounding_default" />
</PreferenceScreen>
```

Some attributes for all Preference elements

- `key`
- `title`
- `summary`
- `defaultValue`

Some attributes for a ListPreference element

- `dialogTitle`
- `entries`
- `entryValues`

The SettingsFragment class

```
package com.murach.tipcalculator;

import android.os.Bundle;
import android.preference.PreferenceFragment;

public class SettingsFragment
        extends PreferenceFragment {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // Load the preferences from an XML resource
        addPreferencesFromResource(R.xml.preferences);
    }
}
```

The SettingsActivity class

```
package com.murach.tipcalculator;

import android.app.Activity;
import android.os.Bundle;

public class SettingsActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // Display the fragment as the main content
        getFragmentManager()
                .beginTransaction()
                .replace(android.R.id.content,
                         new SettingsFragment())
                .commit();
    }
}
```

How to work with preferences

Step 1: Define the instance variables

```
private SharedPreferences prefs;
private boolean rememberTipPercent = true;
private int rounding = ROUND_NONE;
```

Step 2: Set the default values (onCreate)

```
PreferenceManager.setDefaultValues(
    this, R.xml.preferences, false);
```

Step 3: Get the SharedPreferences object (onCreate)

```
prefs = PreferenceManager.getDefaultSharedPreferences(this);
```

Step 4: Get the preferences (onResume)

```
rememberTipPercent = prefs.getBoolean(
    "pref_remember_percent", true);
rounding = Integer.parseInt(
    prefs.getString("pref_rounding", "0"));
```

Some methods of the SharedPreferences object

```
getBoolean(key, default)
getString(key, default)
getInt(key, default)
getLong(key, default)
getFloat(key, default)
```

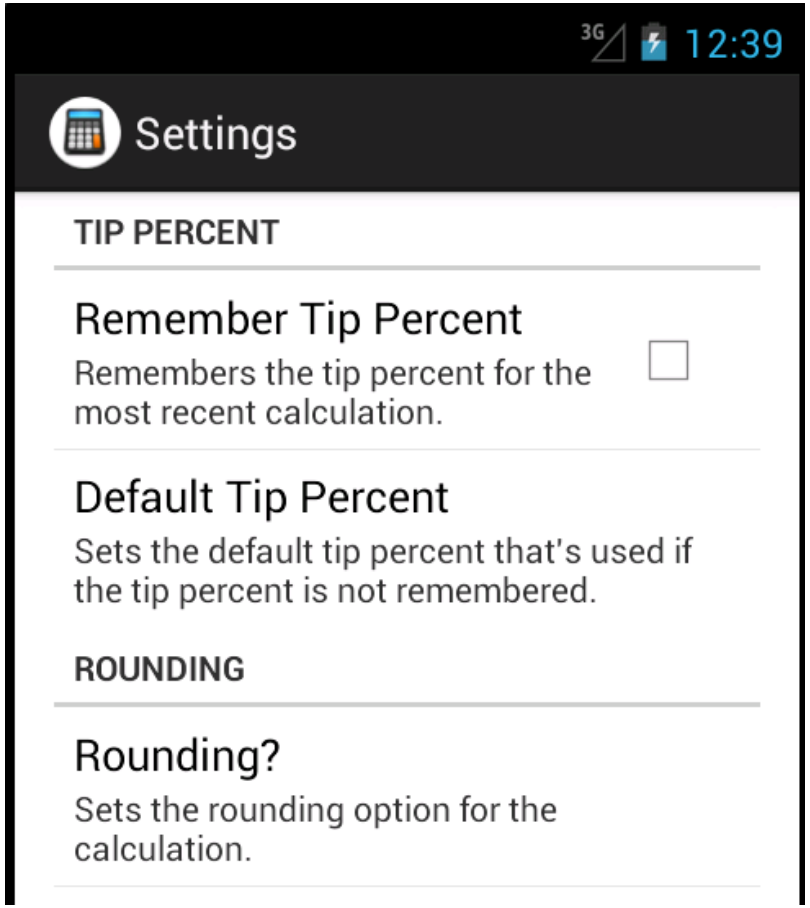
Use the “Remember Tip Percent” preference in the onResume method

```
if (rememberTipPercent) {
    tipPercent = prefs.getFloat("tipPercent", 0.15f);
}
else {
    tipPercent = 0.15f;
}
```

Use the “Rounding” preference in the calculateAndDisplay method

```
float tipPercentToDisplay = 0;
if (rounding == ROUND_NONE) {
    tipAmount = billAmount * tipPercent;
    totalAmount = billAmount + tipAmount;
    tipPercentToDisplay = tipPercent;
}
else if (rounding == ROUND_TIP) {
    tipAmount = StrictMath.round(billAmount * tipPercent);
    totalAmount = billAmount + tipAmount;
    tipPercentToDisplay = tipAmount / billAmount;
}
else if (rounding == ROUND_TOTAL) {
    float tipNotRounded = billAmount * tipPercent;
    totalAmount = StrictMath.round(
        billAmount + tipNotRounded);
    tipAmount = totalAmount - billAmount;
    tipPercentToDisplay = tipAmount / billAmount;
}
```

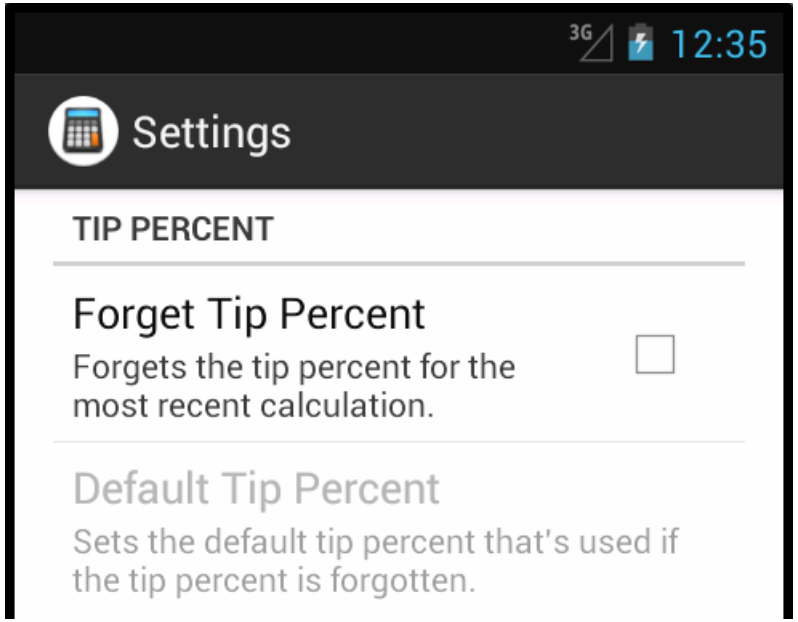
A Settings activity that uses categories



The XML for the preferences

```
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen
    xmlns:android="http://schemas.android.com/apk/res/android">
    <PreferenceCategory
        android:title="@string/percent_category_title"
        android:key="pref_percent_category">
        <CheckBoxPreference
            android:key="pref_remember_percent"
            android:title="@string/forget_remember_title"
            android:summary="@string/forget_remember_summary"
            android:defaultValue="false" />
        <ListPreference
            android:key="pref_default_percent"
            android:title="@string/default_percent_title"
            android:summary="@string/default_percent_summary"
            android:dialogTitle="@string/default_percent_title"
            android:entries="@array/default_percent_keys"
            android:entryValues="@array/default_percent_values"
            android:defaultValue=
                "@string/default_percent_default" />
    </PreferenceCategory>
    ...
</PreferenceScreen>
```

A Settings activity that uses dependencies



The dependency attribute

`dependency`

The XML for the preferences

```
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen
xmlns:android="http://schemas.android.com/apk/res/android">
    <PreferenceCategory
        android:title="@string/percent_category_title"
        android:key="pref_percent_category">
        <CheckBoxPreference
            android:key="pref_forget_percent"
            android:title="@string/forget_percent_title"
            android:summary="@string/forget_percent_summary"
            android:defaultValue="false" />
        <ListPreference
            android:key="pref_default_percent"
            android:title="@string/default_percent_title"
            android:summary="@string/default_percent_summary"
            android:dependency="pref_forget_percent"
            android:dialogTitle="@string/default_percent_title"
            android:entries="@array/default_percent_keys"
            android:entryValues="@array/default_percent_values"
            android:defaultValue="
                @string/default_percent_default" />
    </PreferenceCategory>
    ...
</PreferenceScreen>
```

A class that works with preferences

```
package com.murach.tipcalculator;

import android.content.SharedPreferences;
import android.content.SharedPreferences.
    OnSharedPreferenceChangeListener;
import android.os.Bundle;
import android.preference.Preference;
import android.preference.PreferenceFragment;
import android.preference.PreferenceManager;

public class SettingsFragment extends PreferenceFragment
implements OnSharedPreferenceChangeListener {

    private SharedPreferences prefs;
    private boolean rememberPercent;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        addPreferencesFromResource(R.xml.preferences);
        prefs = PreferenceManager.getDefaultSharedPreferences
            (getActivity());
    }
```

A class that works with preferences (continued)

```
@Override
public void onResume() {
    super.onResume();
    rememberPercent = prefs.getBoolean(
        "pref_remember_percent", true);
    this.setDefaultPercentPreference(rememberPercent);
    prefs.registerOnSharedPreferenceChangeListener(this);
}

private void setDefaultPercentPreference(
    boolean rememberPercent) {
    Preference defaultPercent =
        findPreference("pref_default_percent");
    if (rememberPercent) {
       defaultPercent.setEnabled(false);
    } else {
        defaultPercent.setEnabled(true);
    }
}
@Override
public void onPause() {
    prefs.unregisterOnSharedPreferenceChangeListener(this);
    super.onPause();
}
```

A class that works with preferences (continued)

```
    @Override
    public void onSharedPreferenceChanged(SharedPreferences prefs,
            String key) {
        if (key.equals("pref_remember_percent")) {
            rememberPercent = prefs.getBoolean(key, true);
        }
        this.setDefaultPercentPreference(rememberPercent);
    }
}
```