Fragments

L9 – Essential Android Skills

Ref: Chapter 9 Murach's Android Programming 2Ed

Learning Outcomes

Applied

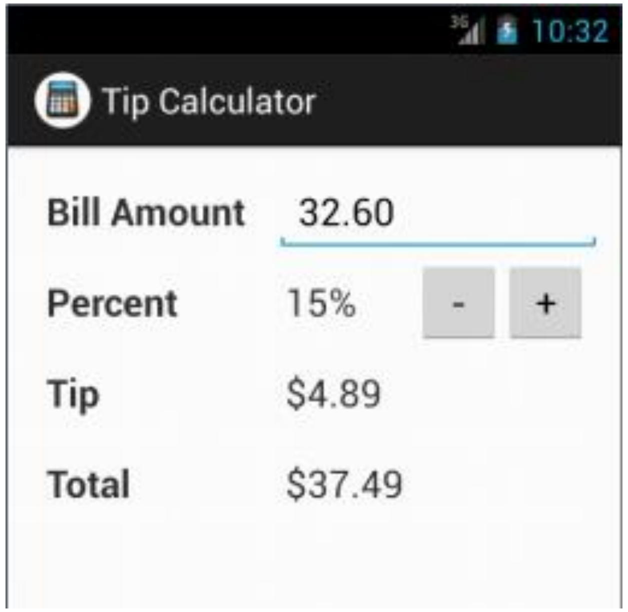
- Create a fragment and display it in an activity.
- Create a preference fragment and display it in an activity.
- Use two or more fragments in a single-pane layout for devices that have small screens.
- Use two or more fragments in a multi-pane layout for devices that have large screens.
- Detect the screen size and display an appropriate single-pane or multi-pane layout.
- Control when the soft keyboard is displayed and the action button that’s displayed.
- Use Java code to replace one fragment with another.

Learning Outcomes

Knowledge

- Describe how you can use fragments in a single-pane or a multi-pane layout.
- Compare the lifecycle methods of a fragment to the lifecycle methods of an activity.
- Describe when the onCreateView and onDestroyView methods of a fragment are called.
- Describe how an alias for a layout can help you create multi-pane layouts.
- Describe how the smallest-width qualifier allows you to detect the screen size.

Two activities displaying two fragments


10:32

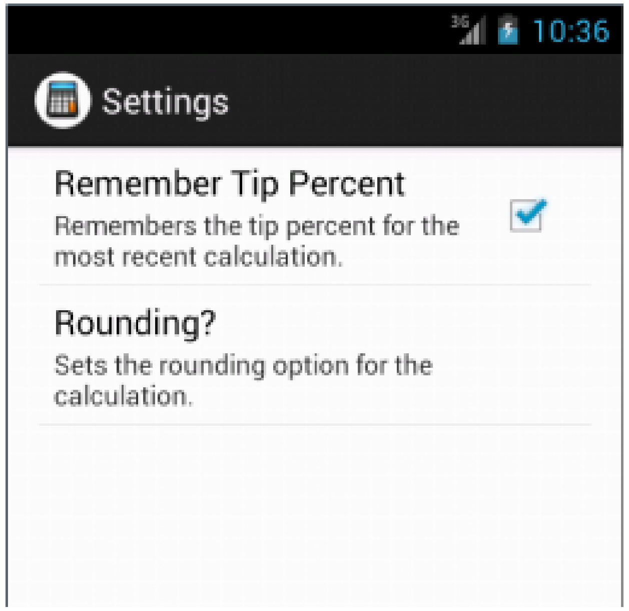
Tip Calculator

Bill Amount 32.60

Percent 15% - +

Tip $4.89

Total $37.49


10:36

Settings

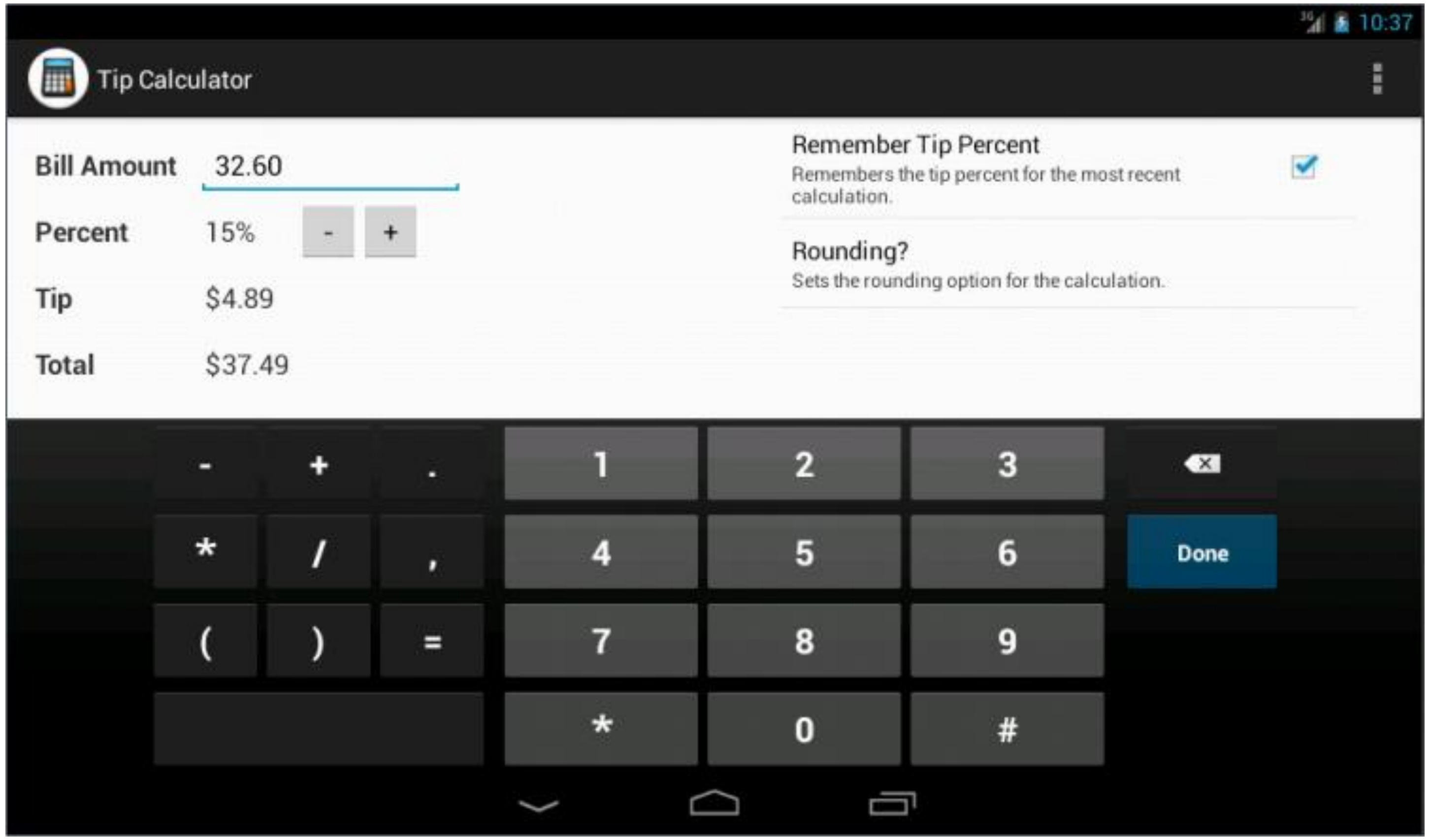
Remember Tip Percent ✓

Remembers the tip percent for the most recent calculation.

Rounding?

Sets the rounding option for the calculation.

One activity displaying two fragments in landscape orientation


Tip Calculator

Bill Amount 32.60

Percent 15%

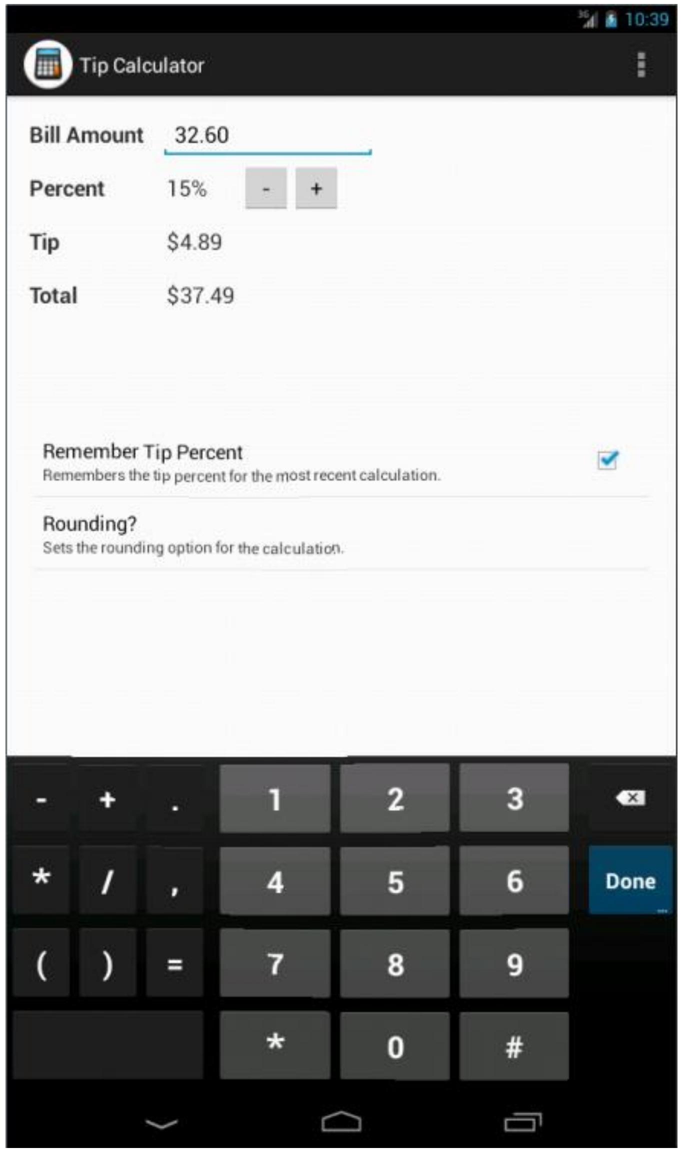
Tip $4.89

Total $37.49

Remember Tip Percent
Remembers the tip percent for the most recent calculation. ✓

Rounding?
Sets the rounding option for the calculation.

One activity displaying two fragments in portrait orientation



Bill Amount 32.60

Percent 15% - +

Tip $4.89

Total $37.49

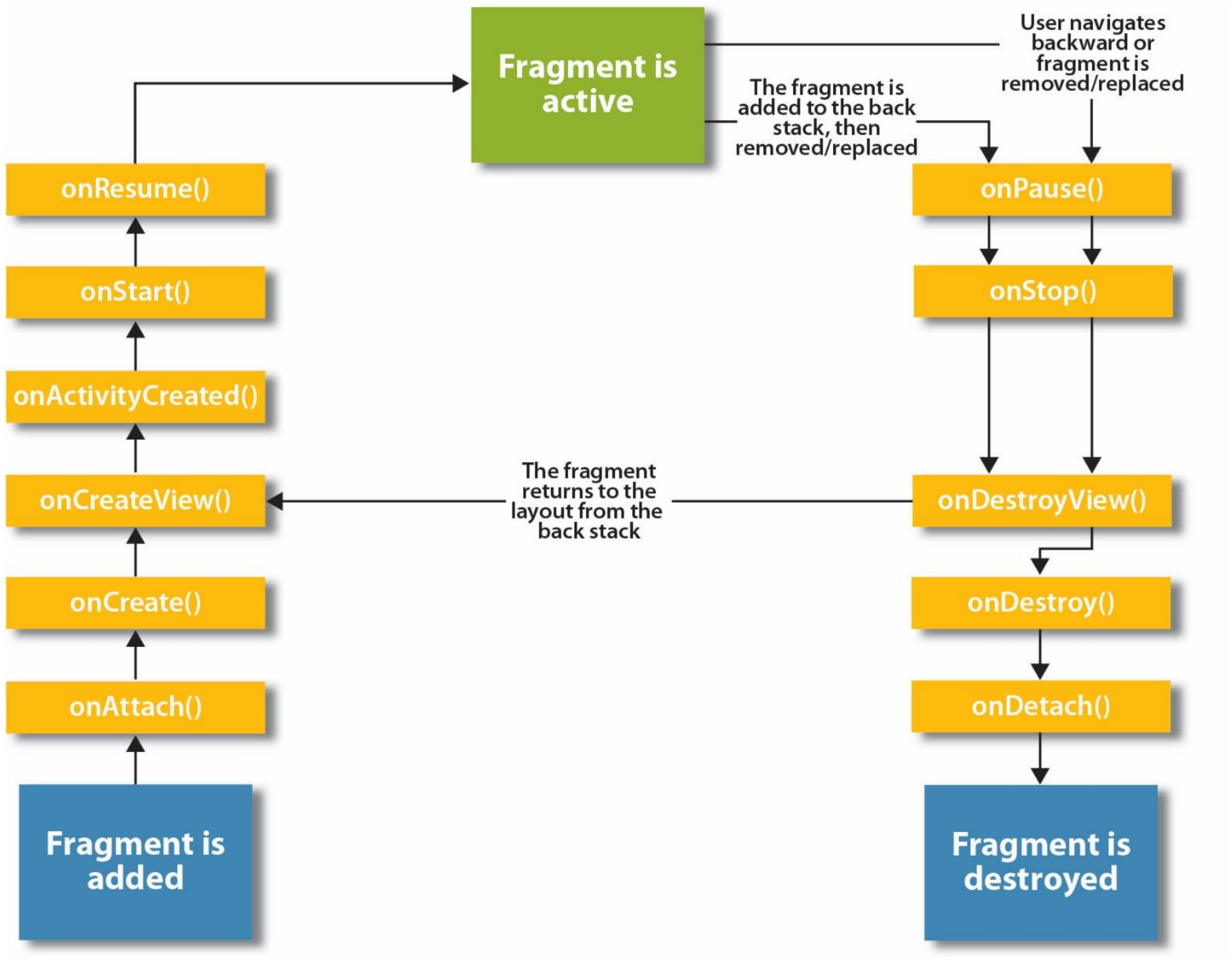
Remember Tip Percent
Remembers the tip percent for the most recent calculation. ✓

Rounding?
Sets the rounding option for the calculation.

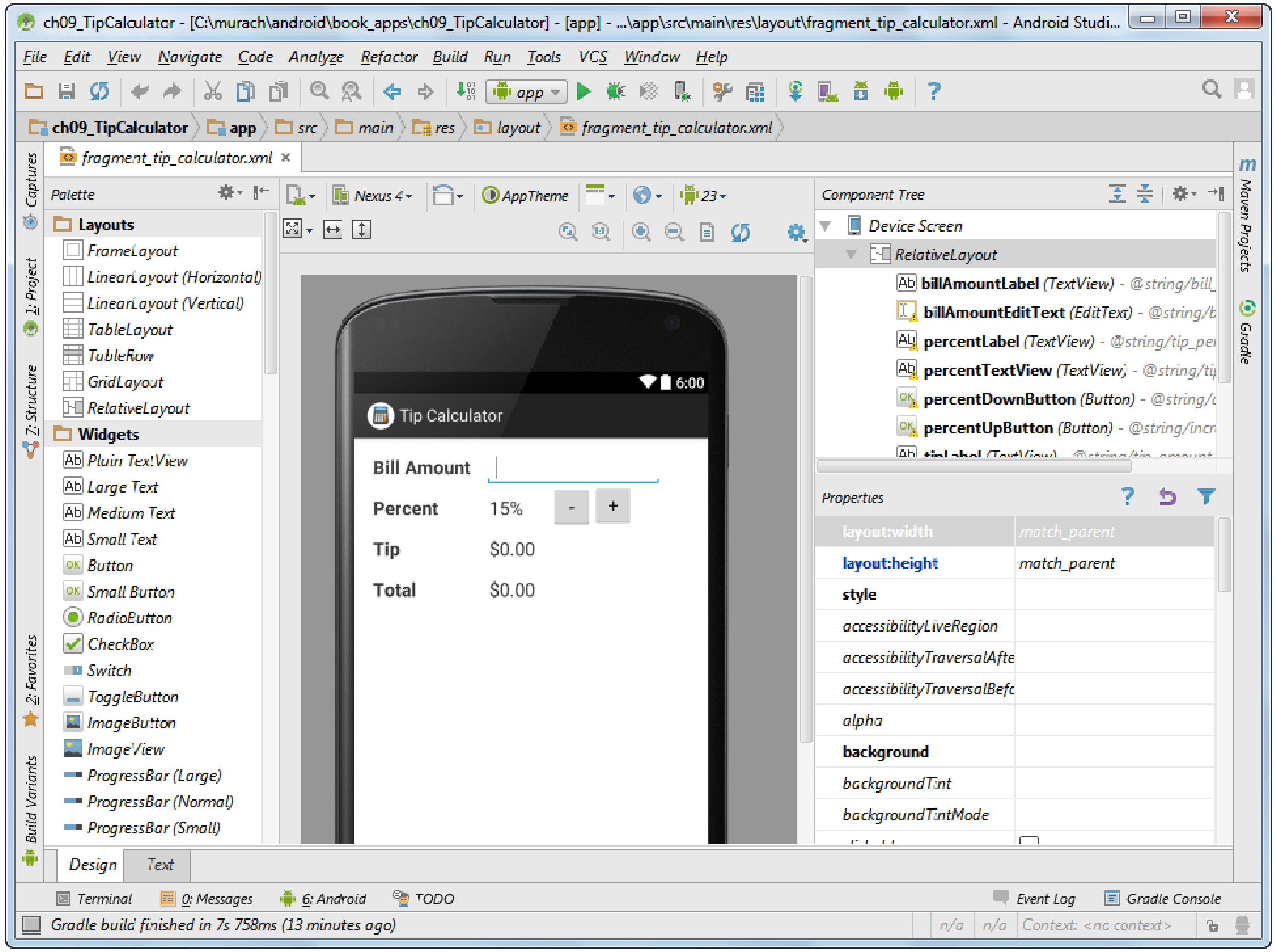
Fragment terminology

- You can use a *fragment* to define part of the user interface for an activity.
- A fragment can be thought of as a *pane*.
- On a small screen, an activity typically only displays a single fragment. This is known as a *single-pane layout*.
- On a large screen, an activity can display multiple fragments. This is known as a *multi-pane layout*.
- A fragment can be reused in multiple activities.

The lifecycle methods of a fragment



The layout for a fragment


ch09_TipCalculator - [C:\murach\android\book_apps\ch09_TipCalculator] - [app] - ...\app\src\main\res\layout\fragment_tip_calculator.xml - Android Studi...

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

ch09_TipCalculator > app > src > main > res > layout > fragment_tip_calculator.xml

fragment_tip_calculator.xml

Palette

Layouts: FrameLayout, LinearLayout (Horizontal), LinearLayout (Vertical), TableLayout, TableRow, GridLayout, RelativeLayout

Widgets: Plain TextView, Large Text, Medium Text, Small Text, Button, Small Button, RadioButton, CheckBox, Switch, ToggleButton, ImageButton, ImageView, ProgressBar (Large), ProgressBar (Normal), ProgressBar (Small)

Nexus 4 | AppTheme | 23

Tip Calculator

Bill Amount	
Percent	15% - +
Tip	$0.00
Total	$0.00

Component Tree

- Device Screen
  - RelativeLayout
    - billAmountLabel (TextView) - @string/bill_
    - billAmountEditText (EditText) - @string/b
    - percentLabel (TextView) - @string/tip_per
    - percentTextView (TextView) - @string/ti
    - percentDownButton (Button) - @string/d
    - percentUpButton (Button) - @string/incr
    - tipLabel (TextView) - @string/tip_amount

Properties

layout:width	match_parent
layout:height	match_parent
style	
accessibilityLiveRegion	
accessibilityTraversalAfte	
accessibilityTraversalBefc	
alpha	
background	
backgroundTint	
backgroundTintMode	

Design | Text

Terminal | 0: Messages | 6: Android | TODO | Event Log | Gradle Console

Gradle build finished in 7s 758ms (13 minutes ago)   n/a n/a Context: <no context>

Method of the Fragment class

```
setHasOptionsMenu(bool)
```

The TipCalculatorFragment class

The declaration

```
public class TipCalculatorFragment extends Fragment
implements OnEditorActionListener, OnClickListener
```

The onCreate method

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

    // set the default values for the preferences
    PreferenceManager.setDefaultValues(getActivity(),
        R.xml.preferences, false);

    // get default SharedPreferences object
    prefs = PreferenceManager.getDefaultSharedPreferences(
        getActivity());

    // turn on the options menu
    setHasOptionsMenu(true);
}
```

The TipCalculatorFragment class (continued)

The onCreateView method

```
@Override
public View onCreateView(LayoutInflater inflater,
        ViewGroup container, Bundle savedInstanceState) {

    // inflate the layout for this fragment
    View view = inflater.inflate(
        R.layout.fragment_tip_calculator, container, false);

    // get references to the widgets
    billAmountEditText = (EditText)
        view.findViewById(R.id.billAmountEditText);

    // set the listeners
    billAmountEditText.setOnEditorActionListener(this);

    // return the View for the layout
    return view;
}
```

The activity_main.xml file

```
<LinearLayout
    xmlns:android=
        "http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <fragment
        android:name=
             "com.murach.tipcalculator.TipCalculatorFragment"
        android:id="@+id/main_fragment"
        android:layout_weight="1"
        android:layout_width="0dp"
        android:layout_height="match_parent" />

</LinearLayout>
```

The TipCalculatorActivity class

```
package com.murach.tipcalculator;

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

public class TipCalculatorActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

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 activity_settings.xml file

```
<LinearLayout

xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <fragment
        android:name=
            "com.murach.tipcalculator.SettingsFragment"
        android:id="@+id/settings_fragment"
        android:layout_weight="1"
        android:layout_width="0dp"
        android:layout_height="match_parent" />

</LinearLayout>
```

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);

        // set the view for the activity using XML
        setContentView(R.layout.activity_settings);
    }
}
```

Two layout files for the main activity

res\layout\activity_main_twopane_land.xml

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <fragment
        android:name=
                "com.murach.tipcalculator.TipCalculatorFragment"
        android:id="@+id/main_fragment"
        android:layout_weight="1"
        android:layout_width="0dp"
        android:layout_height="match_parent" />

    <fragment
        android:name="com.murach.tipcalculator.SettingsFragment"
        android:id="@+id/settings_fragment"
        android:layout_weight="1"
        android:layout_width="0dp"
        android:layout_height="match_parent" />

</LinearLayout>
```

Two layout files for the main activity (continued)

res\layout\activity_main_twopane_port.xml

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <fragment
        android:name=
                "com.murach.tipcalculator.TipCalculatorFragment"
        android:id="@+id/main_fragment"
        android:layout_weight="1"
        android:layout_height="0dp"
        android:layout_width="match_parent" />

    <fragment
        android:name="com.murach.tipcalculator.SettingsFragment"
        android:id="@+id/settings_fragment"
        android:layout_weight="1"
        android:layout_width="match_parent"
        android:layout_height="0dp" />

</LinearLayout>
```

Smallest-width qualifiers examples

Qualifier	Typical screen size	Typical device
sw480dp	640dp x 480dp	5" tablet
sw600dp	1024dp x 600dp	7" tablet
sw720dp	960dp x 720dp	10" tablet

res\values-sw600dp-land\layout.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <item name="activity_main" type="layout">
        @layout/activity_main_twopane_land
    </item>
</resources>
```

res\values-sw600dp-port\layout.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <item name="activity_main" type="layout">
        @layout/activity_main_twopane_port
    </item>
</resources>
```

Some values for the imeOptions attribute

```
actionDone
actionNext
actionPrevious
actionGo
actionSeach
```

An EditText widget that uses a soft keyboard

```
<EditText
    android:id="@+id/billAmountEditText"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignBaseline="@+id/billAmountLabel"
    android:layout_marginLeft="5dp"
    android:layout_toRightOf="@+id/billAmountLabel"
    android:ems="8"
    android:inputType="numberDecimal"
    android:imeOptions="actionDone"
    android:text="@string/bill_amount"
    android:textSize="20sp" >

    <requestFocus />
</EditText>
```

How to control the soft keyboard

- To hide the soft keyboard when the activity starts, delete the requestFocus element.
- To set the action button for the soft keyboard, use the imeOptions attribute.

Method of the Activity and Fragment classes

`getFragmentManager()`

Method of the FragmentManager class

`findFragmentById(id)`

The onCreateOptionsMenu method of the Tip Calculator fragment

```
@Override
public void onCreateOptionsMenu(Menu menu,
        MenuInflater inflater) {

    // attempt to get the fragment
    SettingsFragment settingsFragment = (SettingsFragment)
            getFragmentManager()
            .findFragmentById(R.id.settings_fragment);

    // if the fragment is null, display the appropriate menu
    if (settingsFragment == null) {
        inflater.inflate(
            R.menu.fragment_tip_calculator, menu);
    } else {
        inflater.inflate(
            R.menu.fragment_tip_calculator_twopane, menu);
    }
}
```

The onSharedPreferenceChanged method of the Settings fragment

```
@Override
public void onSharedPreferenceChanged(
        SharedPreferences prefs, String key) {

    // attempt to get the fragment
    TipCalculatorFragment tipFragment =
        (TipCalculatorFragment) getFragmentManager()
            .findFragmentById(R.id.main_fragment);

    // if the fragment is not null, call a method from it
    if (tipFragment != null) {
        tipFragment.onResume();
    }
}
```

Method of the FragmentManager class

```
beginTransaction()
```

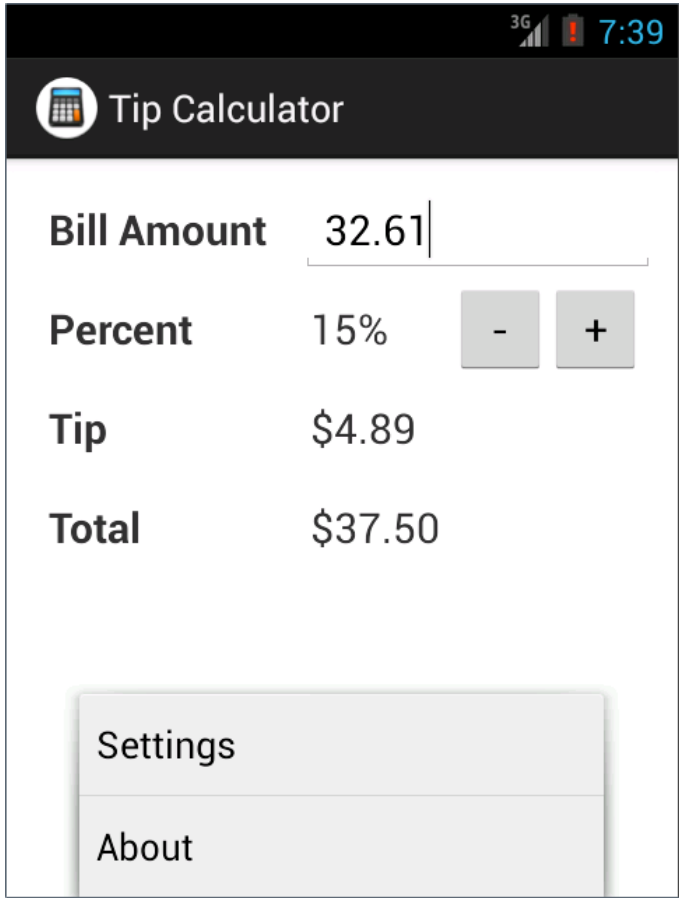
Methods of the FragmentTransaction class

```
replace(id, fragment)
commit()
```

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();
    }
}
```

An activity with an options menu that has two items



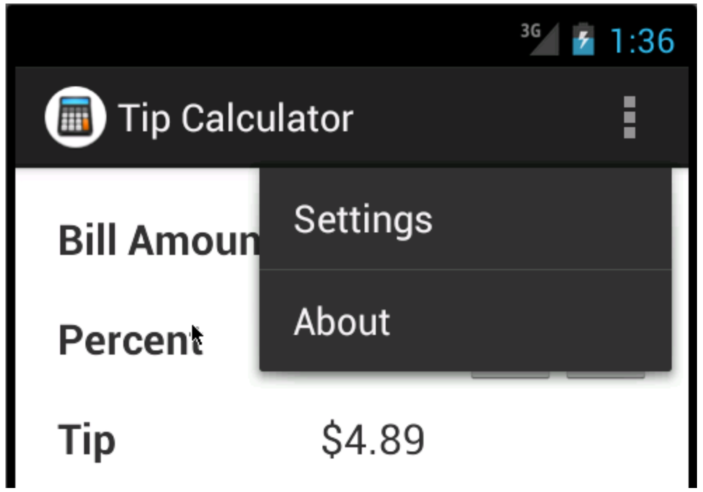
Bill Amount 32.61

Percent 15% - +

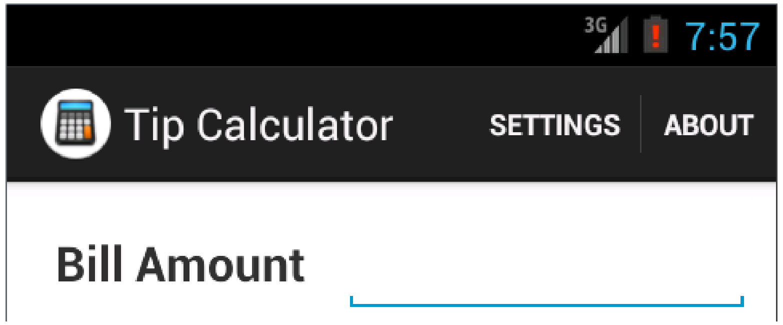
Tip $4.89

Total $37.50

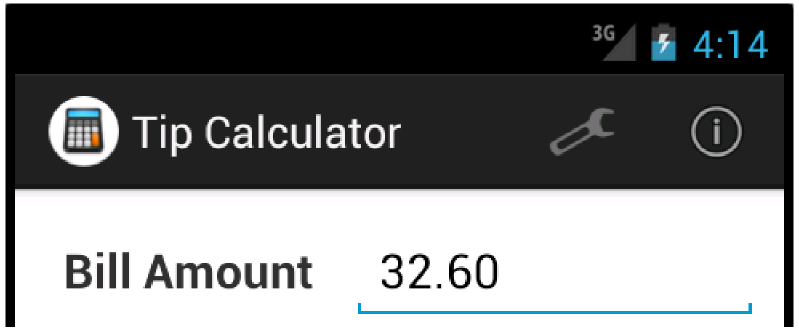
The same options menu displayed from an action overflow icon



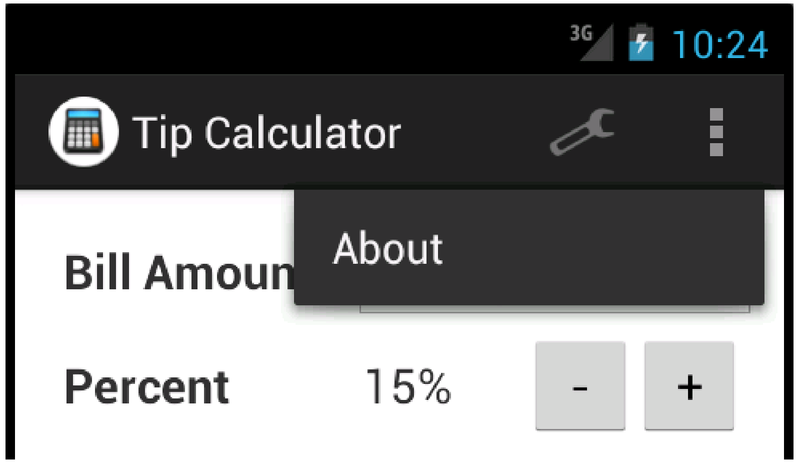
Two action items displayed as text



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);
```

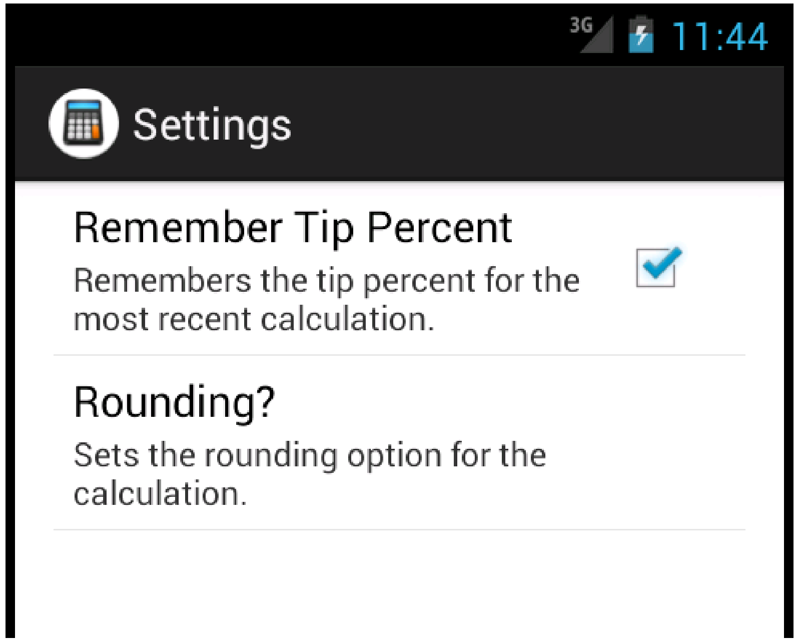
One statement

```
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);
    }
}
```

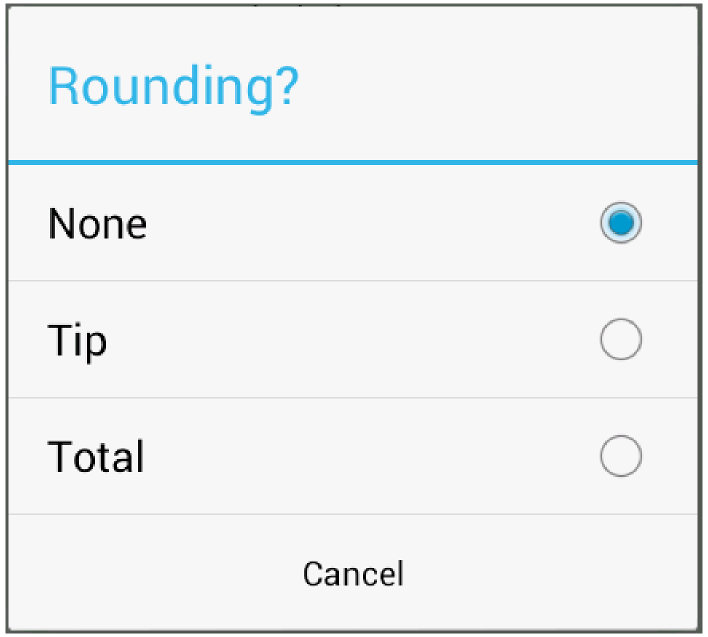
The Settings activity



Remember Tip Percent ✓
Remembers the tip percent for the most recent calculation.

Rounding?
Sets the rounding option for the calculation.

The dialog for the “Rounding?” item


Rounding?

None [✓]

Tip

Total

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)
```

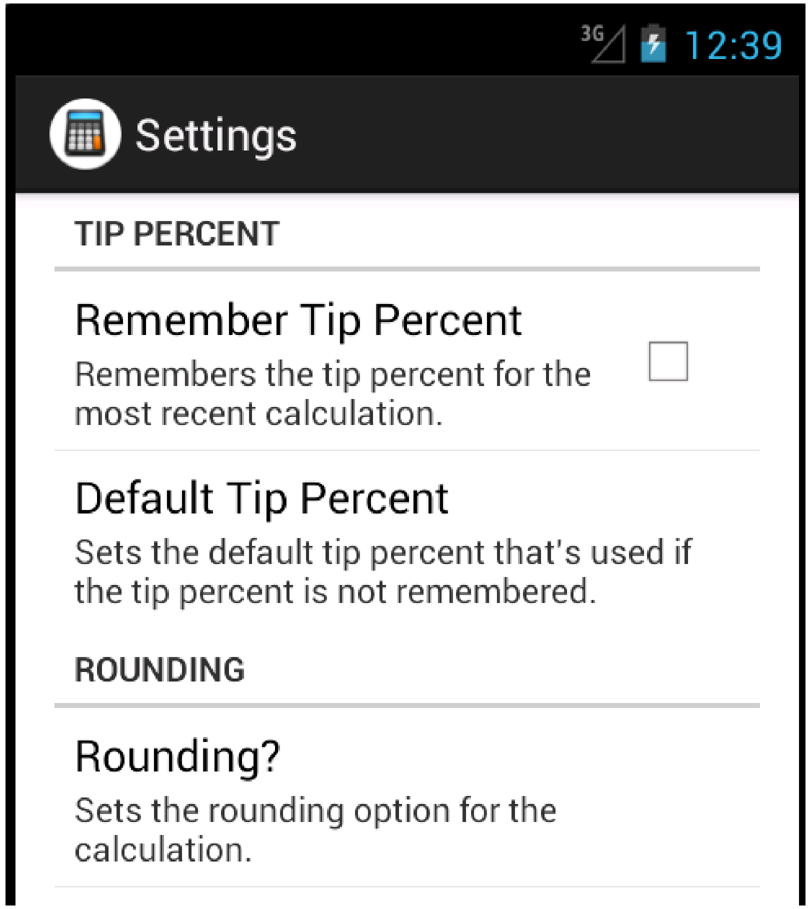
Use the “Remember Tip Percent” preference in the onResume method

```
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



Settings

TIP PERCENT

Remember Tip Percent
Remembers the tip percent for the most recent calculation.

Default Tip Percent
Sets the default tip percent that's used if the tip percent is not remembered.

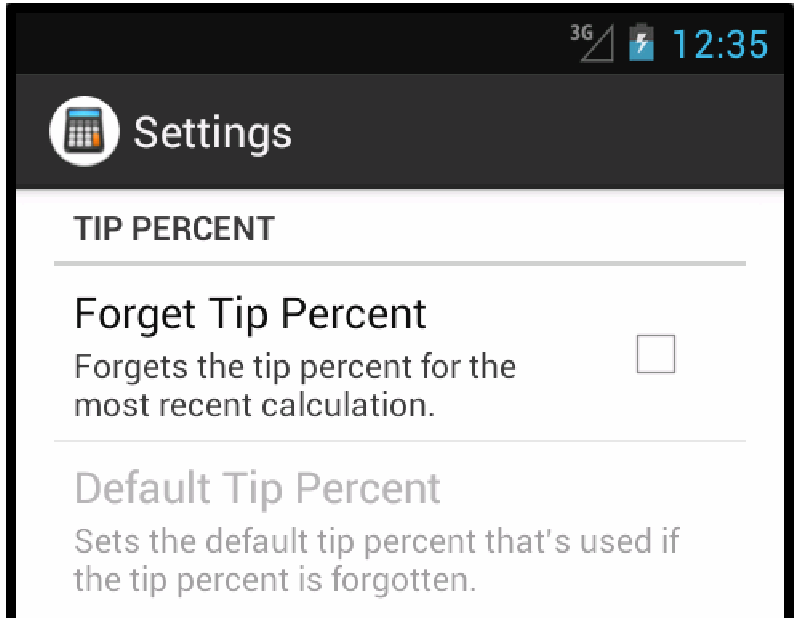
ROUNDING

Rounding?
Sets the rounding option for the calculation.

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



12:35

Settings

TIP PERCENT

Forget Tip Percent

Forgets the tip percent for the most recent calculation.

Default Tip Percent

Sets the default tip percent that's used if the tip percent is forgotten.

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);
    }
}
```