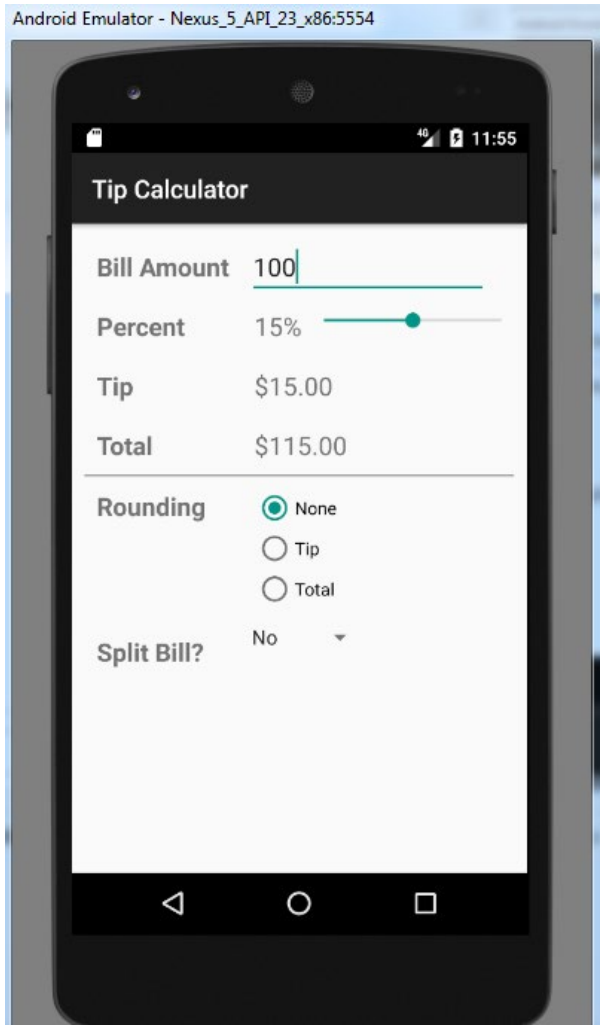
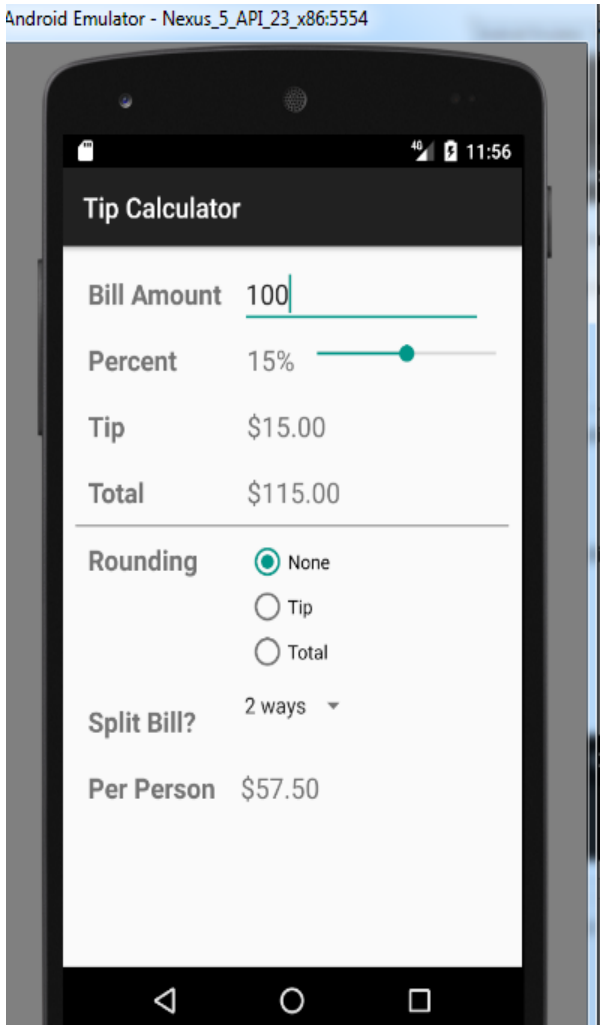
Event Handling

L6 – Essential Android Skills

Ref: Chapter 6 Murach's Android Programming 2Ed

Ch06_TipCalculator





Learning Outcomes

Applied

- Handle events by using the current class as the listener.
- Handle events using an anonymous class or anonymous inner class as the listener.
- Handle events that can occur on check boxes, radio buttons, radio groups, spinners, and seek bars.
- Handle Key events.
- Handle Touch events.

Knowledge

- Distinguish between high-level and low-level events.
- Name the four types of classes that can be used to listen for events and distinguish between each technique.

Listeners for high-level events

Class	Nested Interface	Method(s)
`EditText`	`OnEditorActionListener`	`onEditorAction`
`CompoundButton`	`OnCheckedChangedListener`	`onCheckedChanged`
`RadioGroup`	`OnCheckedChangedListener`	`onCheckedChanged`
`AdapterView`	`OnItemSelectedListener`	`onItemSelected` `onNothingSelected`

`SeekBar`	`OnSeekBarChangeListener`	`onProgressChanged`
		`onStartTrackingTouch`
		`onStopTrackingTouch`

Listeners for low-level events

Class	Nested Interface	Method
`View`	`OnClickListener`	`onClick`
	`OnLongClickListener`	`onLongClick`
	`OnKeyListener`	`onKey`
	`OnFocusChangeListener`	`onFocusChange`
	`OnTouchListener`	`onTouch`

Step 1: Import the interface for the listener

```
import android.view.View.OnClickListener;
```

(Step 1 is the same for all 4 techniques that follow)

Use the current class as the listener

Step 2a: Implement the interface for the listener

```
public class TipCalculatorActivity extends Activity
implements OnClickListener {
```

Step 2b: Implement the interface for the listener

```
@Override
public void onClick(View v) {
    switch (v.getId()) {
        case R.id.percentDownButton:
            tipPercent = tipPercent - .01f;
            calculateAndDisplay();
            break;
        case R.id.percentUpButton:
            tipPercent = tipPercent + .01f;
            calculateAndDisplay();
            break;
    }
}
```

Step 3: Set the listeners

```
percentUpButton.setOnClickListener(this);
percentDownButton.setOnClickListener(this);
```

Use a separate named class as the listener

Step 2: Code a separate class that implements the listener

```
class ButtonListener implements OnClickListener {
    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.percentDownButton:
                tipPercent = tipPercent - .01f;
                calculateAndDisplay();
                break;
            case R.id.percentUpButton:
                tipPercent = tipPercent + .01f;
                calculateAndDisplay();  break;
        }
    }
}
```

Step 3: Create an instance of the listener

```
ButtonListener buttonListener = new ButtonListener();
```

Step 4: Set the listeners

```
percentUpButton.setOnClickListener(buttonListener);
percentDownButton.setOnClickListener(buttonListener);
```

Use an anonymous class as the listener

Step 2: Create an instance variable for the listener

```
private OnClickListener buttonListener = new
    OnClickListener() {
    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.percentDownButton:
                tipPercent = tipPercent - .01f;
                calculateAndDisplay();
                break;
            case R.id.percentUpButton:
                tipPercent = tipPercent + .01f;
                calculateAndDisplay();
                break;
        }
    }
};
```

Step 3: Set the listeners

```
percentUpButton.setOnClickListener(buttonListener);
percentDownButton.setOnClickListener(buttonListener);
```

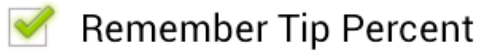
Use an anonymous inner class as the listener

Step 2: Set the listeners and implement the interfaces for the listeners

```
percentUpButton.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        tipPercent = tipPercent + .01f;
        calculateAndDisplay();
    }
});

percentDownButton.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        tipPercent = tipPercent - .01f;
        calculateAndDisplay();
    }
});
```

A check box



A method of the View class

```
getId()
```

An event handler for a check box

```
@Override
public void onCheckedChanged(CompoundButton widget,
        boolean isChecked) {
    switch (widget.getId()) {
        case R.id.rememberPercentCheckBox:
            if (isChecked) {
                rememberTipPercent = true;
            }
            else {
                rememberTipPercent = false;
            }
            break;
    }
}
```

Another event handler for a check box

```
@Override
public void onCheckedChanged(CompoundButton widget,
        boolean isChecked) {
    rememberTipPercent = isChecked;
}
```

Three radio buttons in a group

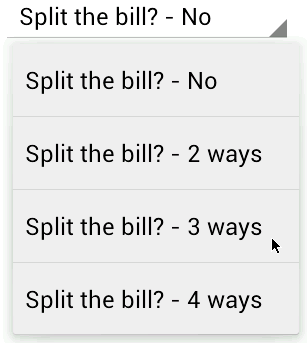
An event handler for a radio group

```
@Override
public void onCheckedChanged(RadioGroup group,
        int checkedId) {
    switch (checkedId) {
        case R.id.noRoundingRadioButton:
            rounding = ROUND_NONE;
            break;
        case R.id.roundTipRadioButton:
            rounding = ROUND_TIP;
            break;
        case R.id.roundTotalRadioButton:
            rounding = ROUND_TOTAL;
            break;
    }
    calculateAndDisplay();
}
```

Another event handler for a radio group

```
@Override
public void onCheckedChanged(RadioGroup group,
        int checkedId) {
    calculateAndDisplay();
}
```

A spinner



An event handler for a spinner

```
@Override
public void onItemSelected(AdapterView<?> parent,
        View v, int position, long id) {
    split = position + 1;
    calculateAndDisplay();
}

@Override
public void onNothingSelected(AdapterView<?> parent) {
    // You typically don't need to include any code here
}
```

A seek bar and a label

An event handler for a seek bar

```
@Override
public void onProgressChanged(SeekBar seekBar,
        int progress, boolean fromUser) {
    percentTextView.setText(progress + "%");
}

@Override
public void onStartTrackingTouch(SeekBar seekBar) {
    // TODO Auto-generated method stub
}

@Override
public void onStopTrackingTouch(SeekBar seekBar) {
    int progress = seekBar.getProgress();
    tipPercent = (float) progress / 100;
    calculateAndDisplay();
}
```

An event handler for the Key event

```
@Override
public boolean onKey(View view,
        int keyCode, KeyEvent event) {
    switch (keyCode) {
        case KeyEvent.KEYCODE_ENTER:
        case KeyEvent.KEYCODE_DPAD_CENTER:

            calculateAndDisplay();

            // hide the soft keyboard
            InputMethodManager imm = (InputMethodManager)
                getSystemService(
                    Context.INPUT_METHOD_SERVICE);
            imm.hideSoftInputFromWindow(
                billAmountEditText.getWindowToken(), 0);

            // consume the event
            return true;
```

An event handler for the Key event (continued)

```
            case KeyEvent.KEYCODE_DPAD_RIGHT:
            case KeyEvent.KEYCODE_DPAD_LEFT:
                if (view.getId() == R.id.percentSeekBar) {
                    calculateAndDisplay();
                }
                break;
        }
        // don't consume the event
        return false;
    }
```

Some constants from the KeyEvent class

```
KEYCODE_ENTER
KEYCODE_DPAD_CENTER
KEYCODE_DPAD_LEFT
KEYCODE_DPAD_RIGHT
KEYCODE_SPACE
```

An event handler for a Touch event

```
@Override
public boolean onTouch(View v, MotionEvent event) {
    float downX, downY, upX, upY;
    int action = event.getAction();
    if (action == MotionEvent.ACTION_DOWN) {
        Log.d("MotionEvent", "ACTION_DOWN");
        downX = event.getX();
        downY = event.getY();
        Log.d("MotionEvent", "downX = " + downX);
        Log.d("MotionEvent", "downY = " + downY);
        return true;
    }
```

An event handler for a Touch event (continued)

```
        else if (action == MotionEvent.ACTION_UP){
            Log.d("MotionEvent", "ACTION_UP");
            upX = event.getX();
            upY = event.getY();
            Log.d("MotionEvent", "upX = " + upX);
            Log.d("MotionEvent", "upY = " + upY);
            return true;
        }
        else {
            return false;
        }
    }
```

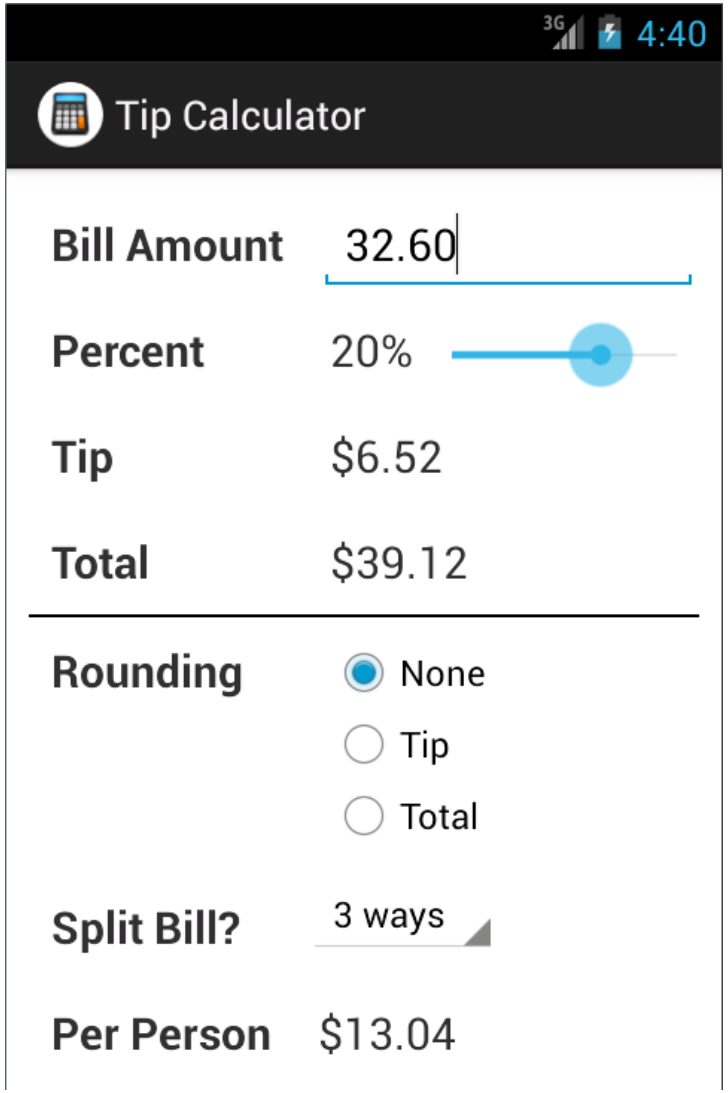
Some constants of the MotionEvent class

```
ACTION_DOWN
ACTION_MOVE
ACTION_UP
```

Some methods of the MotionEvent class

```
getAction()
getX()
getY()
getHistorySize()
getHistoricalX(int i)
getHistoricalY(int i)
```

The user interface



The Java code for the activity

```
package com.murach.tipcalculator;

import java.text.NumberFormat;

import android.os.Bundle;
import android.view.KeyEvent;
import android.view.View;
import android.view.View.OnKeyListener;
import android.view.inputmethod.EditorInfo;
import android.view.inputmethod.InputMethodManager;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemSelectedListener;
import android.widget.ArrayAdapter;
import android.widget.EditText;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.RadioGroup.OnCheckedChangeListener;
import android.widget.SeekBar;
import android.widget.SeekBar.OnSeekBarChangeListener;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.TextView.OnEditorActionListener;
import android.app.Activity;
```

The Java code for the activity (continued)

```
import android.content.Context;
import android.content.SharedPreferences;
import android.content.SharedPreferences.Editor;

public class TipCalculatorActivity extends Activity
    implements OnEditorActionListener, OnSeekBarChangeListener,
        OnCheckedChangeListener, OnItemSelectedListener,
        OnKeyListener {

    // define variables for the widgets
    private EditText billAmountEditText;
    private TextView percentTextView;
    private SeekBar percentSeekBar;
    private TextView tipTextView;
    private TextView totalTextView;
    private RadioGroup roundingRadioGroup;
    private RadioButton roundNoneRadioButton;
    private RadioButton roundTipRadioButton;
    private RadioButton roundTotalRadioButton;
    private Spinner splitSpinner;
    private TextView perPersonLabel;
    private TextView perPersonTextView;
```

The Java code for the activity (continued)

```
// define the SharedPreferences object
private SharedPreferences savedValues;

// define rounding constants
private final int ROUND_NONE = 0;
private final int ROUND_TIP = 1;
private final int ROUND_TOTAL = 2;

// define instance variables
private String billAmountString = "";
private float tipPercent = .15f;
private int rounding = ROUND_NONE;
private int split = 1;
```

The Java code for the activity (continued)

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

    // get references to the widgets
    billAmountEditText = (EditText)
        findViewById(R.id.billAmountEditText);
    percentTextView = (TextView)
        findViewById(R.id.percentTextView);
    percentSeekBar = (SeekBar)
        findViewById(R.id.percentSeekBar);
    tipTextView = (TextView) findViewById(R.id.tipTextView);
    totalTextView = (TextView) findViewById(R.id.totalTextView);
    roundingRadioGroup = (RadioGroup)
        findViewById(R.id.roundingRadioGroup);
    roundNoneRadioButton = (RadioButton)
        findViewById(R.id.roundNoneRadioButton);
    roundTipRadioButton = (RadioButton)
        findViewById(R.id.roundTipRadioButton);
    roundTotalRadioButton = (RadioButton)
        findViewById(R.id.roundTotalRadioButton);
    splitSpinner = (Spinner) findViewById(R.id.splitSpinner);
```

The Java code for the activity (continued)

```
perPersonLabel = (TextView)
    findViewById(R.id.perPersonLabel);
perPersonTextView = (TextView)
    findViewById(R.id.perPersonTextView);

// set array adapter for spinner
ArrayAdapter<CharSequence> adapter =
    ArrayAdapter.createFromResource(
        this, R.array.split_array,
        android.R.layout.simple_spinner_item);
adapter.setDropDownViewResource(
    android.R.layout.simple_spinner_dropdown_item);
splitSpinner.setAdapter(adapter);

// set the listeners
billAmountEditText.setOnEditorActionListener(this);
billAmountEditText.setOnKeyListener(this);
percentSeekBar.setOnSeekBarChangeListener(this);
percentSeekBar.setOnKeyListener(this);
roundingRadioGroup.setOnCheckedChangeListener(this);
roundingRadioGroup.setOnKeyListener(this);
splitSpinner.setOnItemSelectedListener(this);
```

The Java code for the activity (continued)

```
        // get SharedPreferences object
        savedValues = getSharedPreferences("SavedValues",
            MODE_PRIVATE);
    }

    @Override
    public void onPause() {
        // save the instance variables
        Editor editor = savedValues.edit();
        editor.putString("billAmountString", billAmountString);
        editor.putFloat("tipPercent", tipPercent);
        editor.putInt("rounding", rounding);
        editor.putInt("split", split);
        editor.commit();

        super.onPause();
    }
```

The Java code for the activity (continued)

```
@Override
public void onResume() {
    super.onResume();

    // get the instance variables
    billAmountString = savedValues.getString(
        "billAmountString", "");
    tipPercent = savedValues.getFloat("tipPercent", 0.15f);
    rounding = savedValues.getInt("rounding", ROUND_NONE);
    split = savedValues.getInt("split", 1);

    // set the bill amount on its widget
    billAmountEditText.setText(billAmountString);

    // set the tip percent on its widget
    int progress = Math.round(tipPercent * 100);
    percentSeekBar.setProgress(progress);
```

The Java code for the activity (continued)

```
        // set rounding on radio buttons
        // NOTE: this executes the onCheckedChanged method,
        // which executes the calculateAndDisplay method
        if (rounding == ROUND_NONE) {
            roundNoneRadioButton.setChecked(true);
        }
        else if (rounding == ROUND_TIP) {
            roundTipRadioButton.setChecked(true);
        }
        else if (rounding == ROUND_TIP) {
            roundTotalRadioButton.setChecked(true);
        }

        // set split on spinner
        // NOTE: this executes the onItemSelected method,
        // which executes the calculateAndDisplay method
        int position = split - 1;
        splitSpinner.setSelection(position);
    }
```

The Java code for the activity (continued)

```
public void calculateAndDisplay() {
    // get the bill amount
    billAmountString = billAmountEditText.getText().toString();
    float billAmount;
    if (billAmountString.equals("")) {
        billAmount = 0;
    }
    else {
        billAmount = Float.parseFloat(billAmountString);
    }

    // get tip percent
    int progress = percentSeekBar.getProgress();
    tipPercent = (float) progress / 100;
```

The Java code for the activity (continued)

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

The Java code for the activity (continued)

```
        // calculate split amount and show/hide split amount widgets
        float splitAmount = 0;
        if (split == 1) {  // no split - hide widgets
            perPersonLabel.setVisibility(View.GONE);
            perPersonTextView.setVisibility(View.GONE);
        }
        else {             // split - show widgets
            splitAmount = totalAmount / split;
            perPersonLabel.setVisibility(View.VISIBLE);
            perPersonTextView.setVisibility(View.VISIBLE);
        }

        // display the results with formatting
        NumberFormat currency = NumberFormat.getCurrencyInstance();
        tipTextView.setText(currency.format(tipAmount));
        totalTextView.setText(currency.format(totalAmount));
        perPersonTextView.setText(currency.format(splitAmount));

        NumberFormat percent = NumberFormat.getPercentInstance();
        percentTextView.setText(percent.format(tipPercent));
    }
```

The Java code for the activity (continued)

```
//*****************************************************
// Event handler for the EditText
//*****************************************************
@Override
public boolean onEditorAction(TextView v,
        int actionId, KeyEvent event) {
    if (actionId == EditorInfo.IME_ACTION_DONE ||
        actionId == EditorInfo.IME_ACTION_UNSPECIFIED) {
        calculateAndDisplay();
    }
    return false;
}
```

The Java code for the activity (continued)

```
//*****************************************************
// Event handler for the SeekBar
//*****************************************************
@Override
public void onStartTrackingTouch(SeekBar seekBar) {
    // TODO Auto-generated method stub
}

@Override
public void onProgressChanged(SeekBar seekBar, int progress,
        boolean fromUser) {
    percentTextView.setText(progress + "%");
}

@Override
public void onStopTrackingTouch(SeekBar seekBar) {
    calculateAndDisplay();
}
```

The Java code for the activity (continued)

```
//*****************************************************
// Event handler for the RadioGroup
//*****************************************************
@Override
public void onCheckedChanged(RadioGroup group, int checkedId) {
    switch (checkedId) {
        case R.id.roundNoneRadioButton:
            rounding = ROUND_NONE;
            break;
        case R.id.roundTipRadioButton:
            rounding = ROUND_TIP;
            break;
        case R.id.roundTotalRadioButton:
            rounding = ROUND_TOTAL;
            break;
    }
    calculateAndDisplay();
}
```

The Java code for the activity (continued)

```
//*****************************************************
// Event handler for the Spinner
//*****************************************************
@Override
public void onItemSelected(AdapterView<?> parent,
        View v, int position, long id) {
    split = position + 1;
    calculateAndDisplay();
}

@Override
public void onNothingSelected(AdapterView<?> parent) {
    // Do nothing
}
```

The Java code for the activity (continued)

```
//*****************************************************
// Event handler for the keyboard and DPad
//*****************************************************
@Override
public boolean onKey(View view, int keyCode, KeyEvent event) {
    switch (keyCode) {
        case KeyEvent.KEYCODE_ENTER:
        case KeyEvent.KEYCODE_DPAD_CENTER:

            calculateAndDisplay();

            // hide the soft keyboard
            InputMethodManager imm = (InputMethodManager)
                    getSystemService(
                        Context.INPUT_METHOD_SERVICE);
            imm.hideSoftInputFromWindow(
                    billAmountEditText.getWindowToken(), 0);

            // consume the event
            return true;
```

The Java code for the activity (continued)

```
            case KeyEvent.KEYCODE_DPAD_RIGHT:
            case KeyEvent.KEYCODE_DPAD_LEFT:
                if (view.getId() == R.id.percentSeekBar) {
                    calculateAndDisplay();
                }
                break;
        }
        // don't consume the event
        return false;
    }
}
```