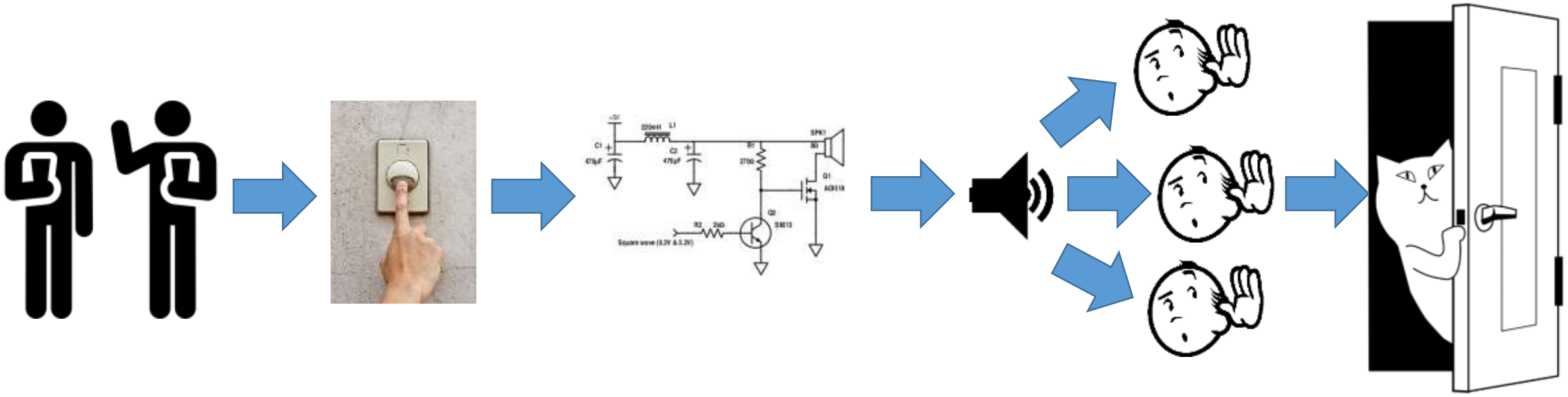
Pemrograman Pola Antar Muka Pengguna (Input Control)

Guntur Maulana Zamroni
Teknik Informatika
Universitas Ahmad Dahlan

View & ViewGroup

- View (child) adalah sebuah objek di layar dimana pengguna dapat berinteraksi dengannya.
- ViewGroup (parents) adalah obyek yang memegang view lain (dan ViewGroup) berguna untuk menentukan tata letak antarmuka pengguna.

Apa itu event?



Event Source

Event Listener

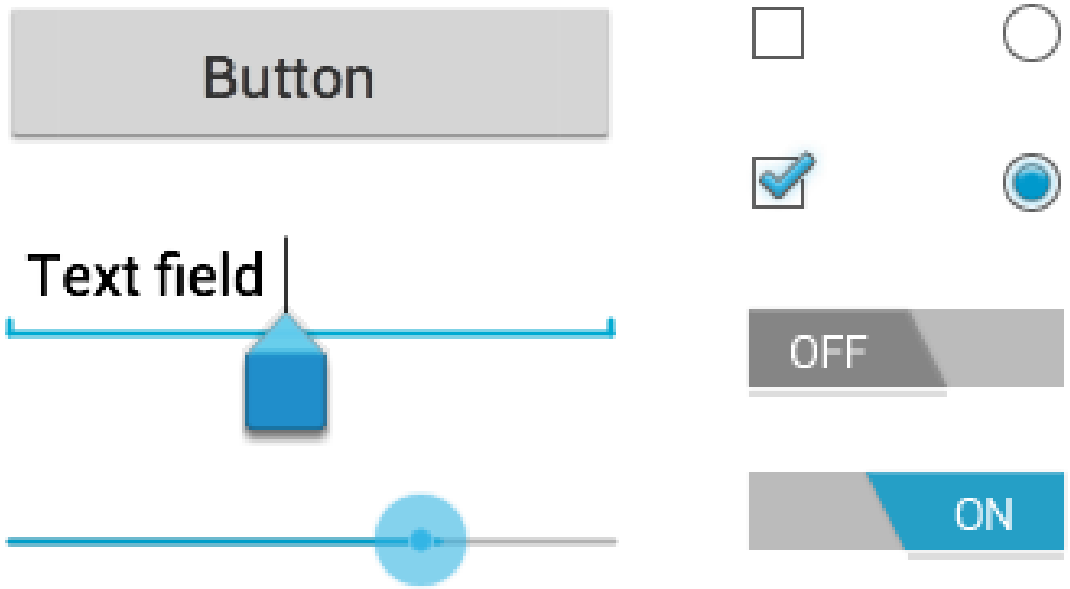
Event Handle

Cara memanfaatkan layout yang kita buat menjadi suatu fungsi tertentu dengan event-event. Terdapat 3 jenis event yaitu :

1. **Event Source** ➔ Merupakan sumber utama dari suatu kejadian yang dilakukan oleh user. misalnya user berinteraksi dengan suatu tombol / button
2. **Event Listener** ➔ Suatu event yang berfungsi menangkap kejadian yang berhubungan dengan user. misalnya user menekan tombol(SetOnClick)
3. **Event Handle** ➔ Setelah user melakukan suatu kejadian, misalnya menekan tombol, maka apakah langkah selanjutnya yang dilakukan ?? itulah yang dinamakan event Handle. misalnya sistem memunculkan tulisan “Selamat datang” ketika button di klik

Input control

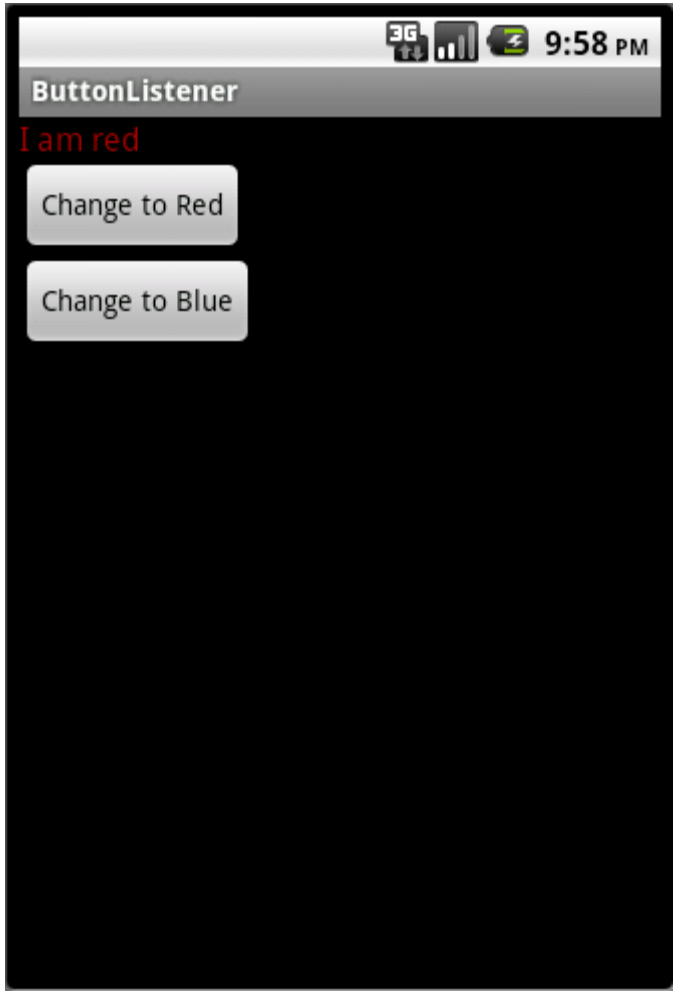
- Input kontrol adalah komponen interaktif didalam antarmuka aplikasi.
- Android menyediakan berbagai macam kontrol yang dapat digunakan dalam UI, seperti buttons, text fields, seek bar, check box, zoom, toggle, dan banyak lagi.



Input Type

- **Input type** adalah untuk mengontrol inputan terhadap widgets **aplikasi android** sehingga inputan sesuai dengan struktur yang diharapkan

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <TextView
        android:id="@+id/txtChange"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="#AA0000"
        android:textSize="16dip"
        android:text="I am red"
        />
    <Button
        android:id="@+id/btnRed"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Change to Red"/>
    <Button
        android:id="@+id/btnBlue"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Change to Blue"/>
</LinearLayout>
```



```java
package com.belajar.button;

import android.app.Activity;
import android.graphics.Color;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

public class main extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        final TextView txtChange = (TextView)findViewById(R.id.txtChange);
        Button btnRed = (Button)findViewById(R.id.btnRed);
        Button btnBlue = (Button)findViewById(R.id.btnBlue);

        btnRed.setOnClickListener(new View.OnClickListener() {

            public void onClick(View v) {
                // TODO Auto-generated method stub
                txtChange.setTextColor(Color.parseColor("#AA0000"));
                txtChange.setText("I am red");
            }
        });

        btnBlue.setOnClickListener(new View.OnClickListener() {

            public void onClick(View v) {
                // TODO Auto-generated method stub
                txtChange.setTextColor(Color.parseColor("#0000AA"));
                txtChange.setText("I am blue");
            }
        });
    }
```

kita membuat fungsi onCreate yaitu fungsi yang akan dijalankan pada saat Activity di create

fungsi setContentView(layout) yang berarti kita “memasang” layout main.xml dalam folder res/layout sebagai tampilan utama Activity ini.

kita “menghubungkan” antar variabel yang kita buat dengan View yang telah kita buat di xml dan kita deklarasikan dalam android:id=”@+id/…..”.

kita memberikan Listener kepada Button untuk menangkap event pada saat Button di klik dengan sintax ‘setOnClickListener’,

kita membuat View.OnClickListener untuk membaca klik atau touch yang dilakukan user pada Button kita

kita memasukkan fungsi yang akan dijalankan di dalam function onClick. dimana onClick mempunyai aksi mengubah warna tulisan TextView dan juga tulisan di dalamnya

Buttons

- A button consists of text or an icon (or both text and an icon) that communicates what action occurs when the user touches it.
- Depending on whether you want a button with text, an icon, or both, you can create the button in your layout in three ways:
  - With text, using the Button class:

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/button_text"
    ... />
```

Buttons

- With an icon, using the ImageButton class:

```
<ImageButton
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/button_icon"
    ... />
```

- With text and an icon, using the Button class with the android:drawableLeft attribute:

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/button_text"
    android:drawableLeft="@drawable/button_icon"
    ... />
```

Using an OnClickListener

- You can also declare the click event handler programmatically rather than in an XML layout.
- This might be necessary if you instantiate the Button at runtime or you need to declare the click behavior in a Fragment subclass.
- To declare the event handler programmatically, create an **View.OnClickListener** object and assign it to the button by calling **setOnClickListener(View.OnClickListener)**.

OnClickListener()

- Event ini disebut juga event listener, suatu event yang berfungsi menangkap kejadian yang berhubungan dengan user.
- Misalnya user menekan tombol (SetOnClick). Event ini sangat berkaitan dengan event handler, setelah user melakukan suatu kejadian, misalnya menekan tombol, maka apakah langkah selanjutnya yang dilakukan ?? Itulah yang dinamakan eventHandler.

Example

```
<Button
    android:id="@+id/button_send"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/button_send"
```

```
Button button = (Button) findViewById(R.id.button_send);
button.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        // Do something in response to button click
    }
});
```

View v, dimana v mengacu pada object yang menerima event klik dari pemakai, dalam hal ini adalah button_send.

- Contoh : tombol button di tekan memunculkan hasil perkalian antara input1 dan input2.
- Nama ID button adalah *button1* ,

  setOnclick adalah klik , maka coding yang berkaitan adalah :

```
tekan=(Button)findViewById(R.id.button1);
tekan.setOnClickListener(new klik());
 class klik implements Button.OnClickListener{
    public void onClick (View v){
          int b1 = Integer.parseInt(bil1.getText().toString());
          int b2 = Integer.parseInt(bil2.getText().toString());

          int hsl = b1 * b2;
          hs.setText(String.valueOf(hsl));
    }
  }
```

Responding to Click Events

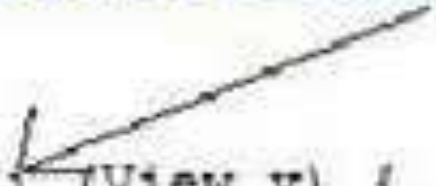
- When the user clicks a button, the Button object receives an on-click event.
- To define the click event handler for a button, add the **android:onClick** attribute to the <Button> element in your XML layout.
- The value for this attribute must be the name of the method you want to call in response to a click event.
- The Activity hosting the layout must then implement the corresponding method.

OnClick()

- Event ini disebut event klasik, dimana sistem kerjanya adalah, tombol button di-klik maka variabel **on click** akan mengeksekusi nama variabel pada strings.xml, kemudian value dari variable strings.xml dianggap sebuah function yang mengeksekusi proses fungsi tersebut
- Prosesnya dapat di gambarkan sebagai berikut:

```
android:onClick="@string/tekankali"
```

```
<string name="tekankali">kali</string>
```



```
public void kali (View v) {
        double a,b,hs;
        a = Double.parseDouble(bil1.getText
().toString());
        b = Double.parseDouble(bil2.getText
().toString());
        hs=a*b ;
        hsk.setText(String.valueOf(hs));
        //hsk.setText(String.valueOf(hs));



    }
```

Responding to Click Events

- For example, here's a layout with a button using android:onClick:

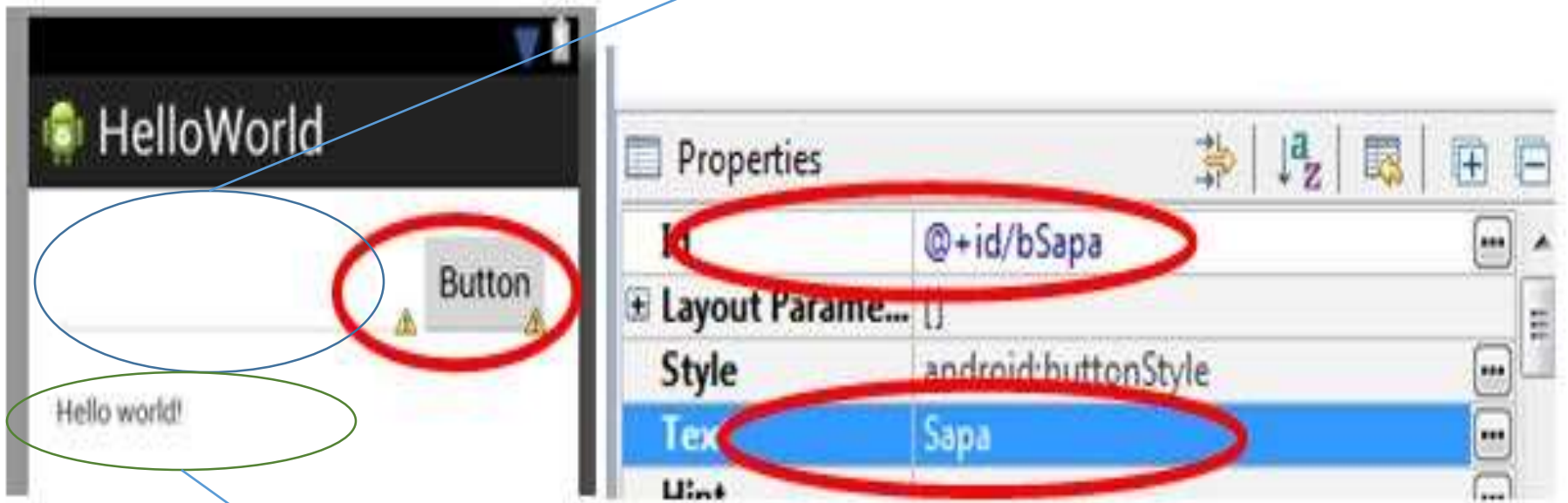
```
<?xml version="1.0" encoding="utf-8"?>
<Button xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/button_send"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/button_send"
    android:onClick="sendMessage" />
```

- following method handles the click event:

```
/** Called when the user touches the button */
public void sendMessage(View view) {
    // Do something in response to button click
}
```

Klik Button kemudian lihat window properties yang berada di sebelah kanan, ganti :
Text dengan “Sapa”
id dengan “bSapa”

id → etNama



id → tvSalam

Sekarang kita akan mengeset agar saat button diklik, method yang diinginkan akan dipanggil. Set atribut android.onClik pada button dengan nama method yang akan menangani event tersebut (code bawah yang di-highlight):

```
1
2   <Button
3   android:id="@+id/bSapa"
4   android:layout_width="wrap_content"
5   android:layout_height="wrap_content"
6   android:layout_alignBottom="@+id/etNama"
7   android:layout_alignParentRight="true"
8   android:onClick="bSapaClick"
    android:text="Sapa" />
```

Penting: setelah update XML, tekan save (ctrl-s). Ini disebabkan file R.java (di direktori /gen) yang berisi semua Id dan digenerate secara otomatis dapat tidak terupdate jika file xml tidak di-save secara eksplisit.

Kemudian buat satu method baru bSapaClick (code dibawah, baris 3-11). **Pastikan nama method sama dengan yang dicantumkan di activity_main.XML**. Nama yang tidak sama akan menyebakan error saat program dijalankan

```
1
2   public class MainActivity extends Activity {
3
4       public void bSapaClick(View v) {
5           //ambil komponen text dan label
6           EditText etNama = (EditText) findViewById(R.id.etNama);
7           TextView tvSalam = (TextView) findViewById(R.id.tvSalam);
8           //ambil masukkan dari user
9           String nama = etNama.getText().toString();
10          //tulis di label
11          tvSalam.setText("Halo "+ nama + " senang bertemu dengan anda");
12      }
```

```
1
2   @Override
3   protected void onCreate(Bundle savedInstanceState) {
4       super.onCreate(savedInstanceState);
5       setContentView(R.layout.activity_main);
6       Button btSapa = (Button) findViewById(R.id.bSapa);
7       btSapa.setOnClickListener(new OnClickListener() {
8           public void onClick(View arg) {
9               EditText etNama = (EditText) findViewById(R.id.etNama);
10              TextView tvSalam = (TextView) findViewById(R.id.tvSalam);
11              String nama = etNama.getText().toString();
12              tvSalam.setText("Halo "+ nama + " senang bertemu dengan anda");
13          }
14      } );
    }
```

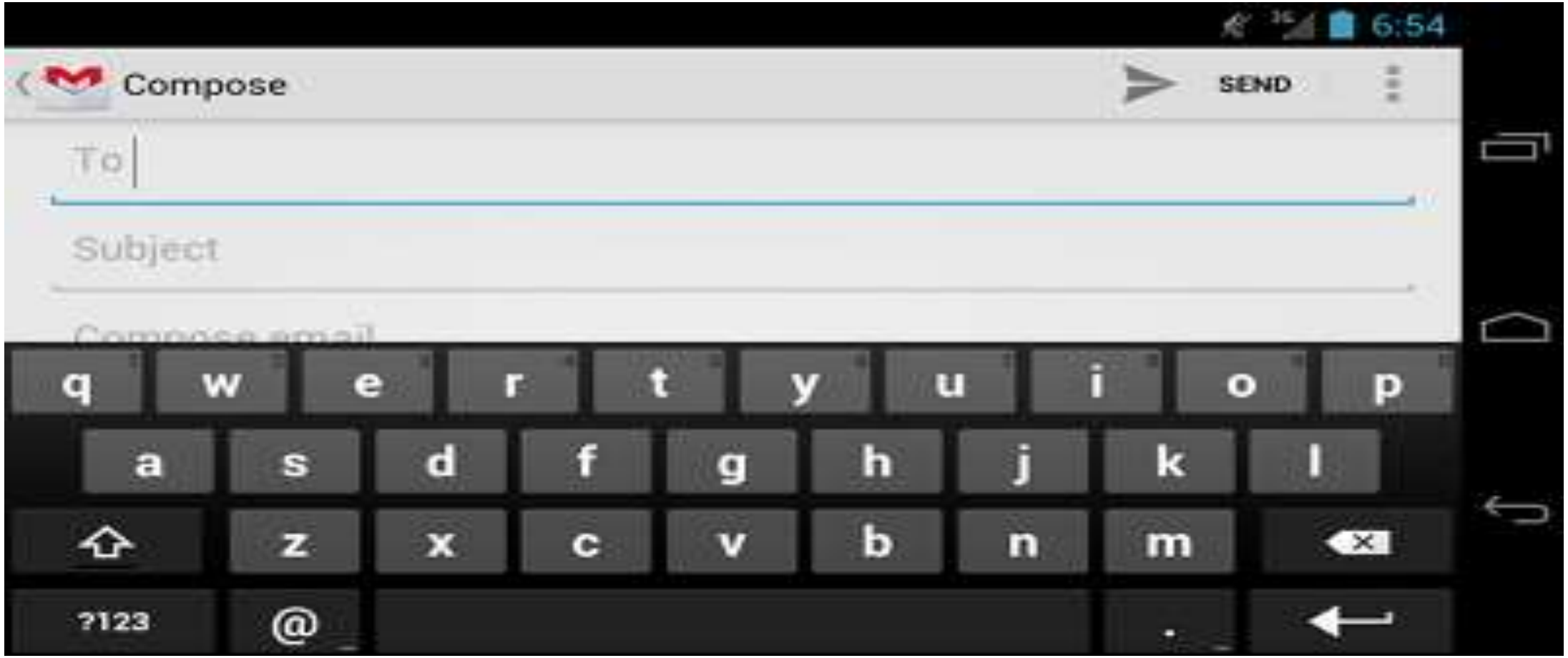
Keterangan :

- **Button btsapa = (Button) findViewById(R.id.bSapa)** merupakan event source yaitu mencari komponen tombol sesuai dengan id yang berada pada file .xml
- **btSapa.setOnClickListener(new OnClickListener( )** merupakan event Listener , yang menangkap kejadian yang dilakukan oleh user. Dalam hal ini user akan menghadapi event onClick yaitu kejadian dimana user mengklik button
- **onClick(View arg)** merupakan **event Handle**. Apa yang akan dilakukan user setelah mengklik tombol akan berada pada method ini. Misal user akan disajikan tampilan teks “Hallo…. senang bertemu dengan anda” dll.

Text Fields

- A text field allows the user to type text into your app. It can be either single line or multi-line.
- Touching a text field places the cursor and automatically displays the keyboard.
- In addition to typing, text fields allow for a variety of other activities, such as text selection (cut, copy, paste) and data look-up via auto-completion.
- You can add a text field to you layout with the EditText object. You should usually do so in your XML layout with a <EditText> element.

Text Fields


6:54
Compose
SEND
To
Subject
Compose email
q w e r t y u i o p
a s d f g h j k l
z x c v b n m
?123
@
.

Specifying the Keyboard Type

- Text fields can have different input types, such as number, date, password, or email address.
- The type determines what kind of characters are allowed inside the field, and may prompt the virtual keyboard to optimize its layout for frequently used characters.
- You can specify the type of keyboard you want for your EditText object with the android:inputType attribute.

Specifying the Keyboard Type

phone

textEmailAddress

number

```
<EditText
    android:id="@+id/email_address"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:hint="@string/email_hint"
    android:inputType="textEmailAddress" />
```

Several different input types

- There are several different input types available for different situations. Here are some of the more common values for **android:inputType**:
  - "text“ → Normal text keyboard.
  - "textEmailAddress“ → Normal text keyboard with the @ character.
  - "textUri“ → Normal text keyboard with the / character.
  - "number“ → Basic number keypad.
  - "phone“ → Phone-style keypad.

Checkboxes

- Checkboxes allow the user to select one or more options from a set.
- Typically, you should present each checkbox option in a vertical list.
- To create each checkbox option, create a CheckBox in your layout.
- Because a set of checkbox options allows the user to select multiple items, **each checkbox is managed separately and you must register a click listener for each one**.

Checkboxes

Sync Browser
5/31/2012 4:58 PM
Sync Calendar
6/1/2012 11:15 AM
Sync Contacts
6/1/2012 3:50 PM

Responding to Click Events

- When the user selects a checkbox, the CheckBox object receives an on-click event.
- To define the click event handler for a checkbox, add the **android:onClick** attribute to the <**CheckBox**> element in your XML layout.
- The value for this attribute must be the name of the method you want to call in response to a click event.
- The Activity hosting the layout must then implement the corresponding method.

XML Example

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <CheckBox android:id="@+id/checkbox_meat"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/meat"
        android:onClick="onCheckboxClicked"/>
    <CheckBox android:id="@+id/checkbox_cheese"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/cheese"
        android:onClick="onCheckboxClicked"/>
</LinearLayout>
```

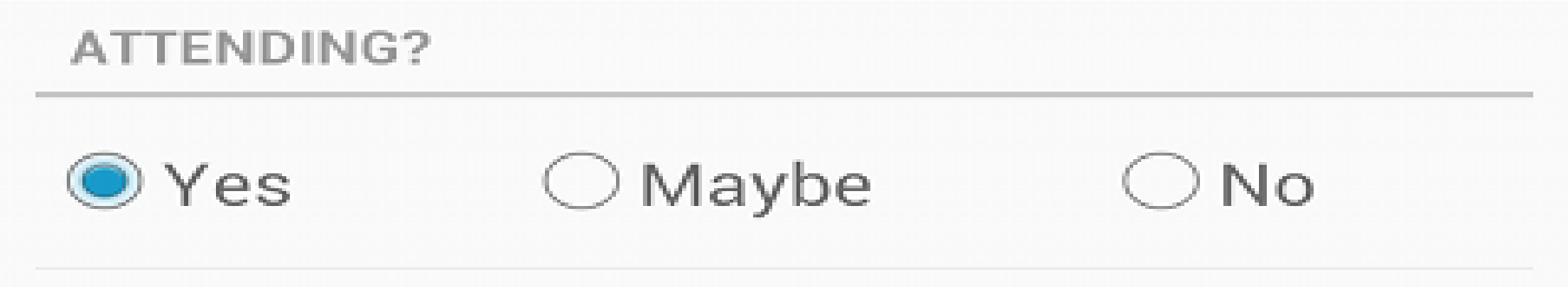
Activity Example

```
public void onCheckboxClicked(View view) {
    // Is the view now checked?
    boolean checked = ((CheckBox) view).isChecked();

    // Check which checkbox was clicked
    switch(view.getId()) {
        case R.id.checkbox_meat:
            if (checked)
                // Put some meat on the sandwich
            else
                // Remove the meat
            break;
        case R.id.checkbox_cheese:
            if (checked)
                // Cheese me
            else
                // I'm lactose intolerant
            break;
        // TODO: Veggie sandwich
    }
}
```

Radio Buttons

- Radio buttons allow the user to select one option from a set.
- You should use radio buttons for optional sets that are mutually exclusive if you think that the user needs to see all available options side-by-side.
- If it's not necessary to show all options side-by-side, use a spinner instead.
- To create each radio button option, create a RadioButton in your layout.
- However, because radio buttons are mutually exclusive, **you must group them together inside a RadioGroup**.
- By grouping them together, the system ensures that only one radio button can be selected at a time.

Radio Buttons



Responding to Click Events

- When the user selects one of the radio buttons, the corresponding RadioButton object receives an on-click event.
- To define the click event handler for a button, add the **android:onClick** attribute to the <**RadioButton**> element in your XML layout.
- The value for this attribute must be the name of the method you want to call in response to a click event.
- The Activity hosting the layout must then implement the corresponding method.

XML Example

```
<?xml version="1.0" encoding="utf-8"?>
<RadioGroup xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">
    <RadioButton android:id="@+id/radio_pirates"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/pirates"
        android:onClick="onRadioButtonClicked"/>
    <RadioButton android:id="@+id/radio_ninjas"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/ninjas"
        android:onClick="onRadioButtonClicked"/>
</RadioGroup>
```

Activity Example

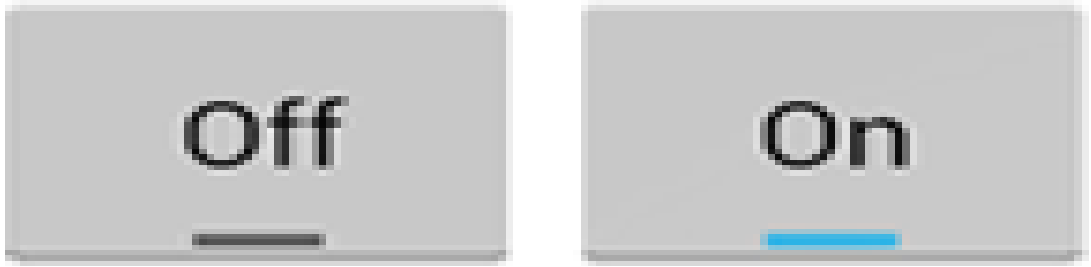
```
public void onRadioButtonClicked(View view) {
    // Is the button now checked?
    boolean checked = ((RadioButton) view).isChecked();

    // Check which radio button was clicked
    switch(view.getId()) {
        case R.id.radio_pirates:
            if (checked)
                // Pirates are the best
            break;
        case R.id.radio_ninjas:
            if (checked)
                // Ninjas rule
            break;
    }
}
```

Toggle Buttons

- A toggle button allows the user to change a setting between two states.
- You can add a basic toggle button to your layout with the ToggleButton object.
- Android 4.0 (API level 14) introduces another kind of toggle button called a switch that provides a slider control, which you can add with a Switch object.
- If you need to change a button's state yourself, you can use the **CompoundButton.setChecked()** or **CompoundButton.toggle()** methods.

Toggle Buttons

Toggle buttons



Switches (in Android 4.0+)

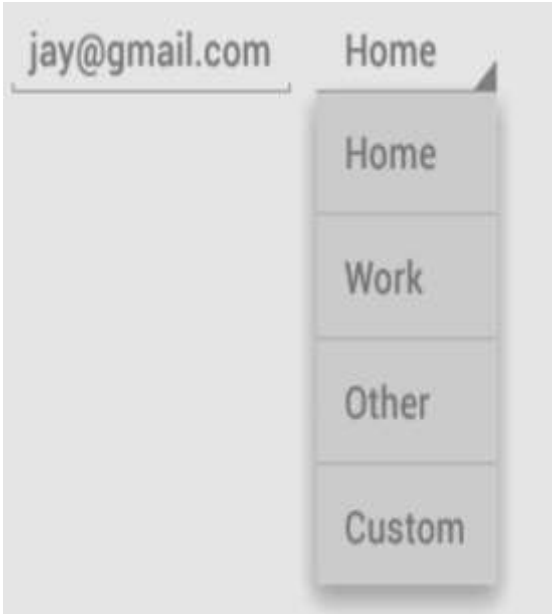
Responding to Button Presses

- To detect when the user activates the button or switch, create an **CompoundButton.OnCheckedChangeListener** object and assign it to the button by calling **setOnCheckedChangeListener()**.
- For example:

```
ToggleButton toggle = (ToggleButton) findViewById(R.id.togglebutton);
toggle.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
    public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
        if (isChecked) {
            // The toggle is enabled
        } else {
            // The toggle is disabled
        }
    }
});
```

Spinners

- Spinners provide a **quick way to select one value from a set**.
- In the default state, a spinner shows its currently selected value.
- Touching the spinner displays a dropdown menu with all other available values, from which the user can select a new one.



Spinners

- You can add a spinner to your layout with the Spinner object. You should usually do so in your XML layout with a <Spinner> element.
- For example:

```
<Spinner
    android:id="@+id/planets_spinner"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" />
```

- To populate the spinner with a list of choices, you then need to specify a **SpinnerAdapter** in your Activity or Fragment source code.

Populate the Spinner with User Choices

- The choices you provide for the spinner can come from any source, but must be provided through a **SpinnerAdapter**, such as an **ArrayAdapter** if the choices are available in an **array** or a **CursorAdapter** if the choices are available from a **database query**.
- For instance, if the available choices for your spinner are pre-determined, you can provide them with a string array defined in a string resource file:

Populate the Spinner with User Choices

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="planets_array">
        <item>Mercury</item>
        <item>Venus</item>
        <item>Earth</item>
        <item>Mars</item>
        <item>Jupiter</item>
        <item>Saturn</item>
        <item>Uranus</item>
        <item>Neptune</item>
    </string-array>
</resources>
```

Populate the Spinner with User Choices

- With an array such as this one, you can use the following code in your Activity or Fragment to supply the spinner with the array using an instance of **ArrayAdapter**:

```
Spinner spinner = (Spinner) findViewById(R.id.spinner);
// Create an ArrayAdapter using the string array and a default spinner layout
ArrayAdapter<CharSequence> adapter = ArrayAdapter.createFromResource(this,
        R.array.planets_array, android.R.layout.simple_spinner_item);
// Specify the layout to use when the list of choices appears
adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
// Apply the adapter to the spinner
spinner.setAdapter(adapter);
```

- The **createFromResource()** method allows you to create an **ArrayAdapter** from the string array.

Populate the Spinner with User Choices

- The third argument for this method is a layout resource that defines how the selected choice appears in the spinner control.
- The simple_spinner_item layout is provided by the platform and is the default layout you should use unless you'd like to define your own layout for the spinner's appearance.
- You should then call setDropDownViewResource(int) to specify the layout the adapter should use to display the list of spinner choices (simple_spinner_dropdown_item is another standard layout defined by the platform).
- Call **setAdapter()** to apply the adapter to your Spinner.

Responding to User Selections

- When the user selects an item from the drop-down, the Spinner object receives an on-item-selected event.
- To define the selection event handler for a spinner, implement the **AdapterView.OnItemSelectedListener** interface and the corresponding **onItemSelected()** callback method.
- For example, here's an implementation of the interface in an Activity:

Responding to User Selections

```
public class SpinnerActivity extends Activity implements OnItemSelectedListener {
    ...

    public void onItemSelected(AdapterView<?> parent, View view,
            int pos, long id) {
        // An item was selected. You can retrieve the selected item using
        // parent.getItemAtPosition(pos)
    }

    public void onNothingSelected(AdapterView<?> parent) {
        // Another interface callback
    }
}
```

Responding to User Selections

- The **AdapterView.OnItemSelectedListener** requires the **onItemSelected( )** and **onNothingSelected( )** callback methods.
- Then you need to specify the interface implementation by calling **setOnItemSelectedListener( )**:

```
Spinner spinner = (Spinner) findViewById(R.id.spinner);
spinner.setOnItemSelectedListener(this);
```

- If you implement the **AdapterView.OnItemSelectedListener** interface with your Activity or Fragment (such as in the example above), you can pass this as the interface instance.

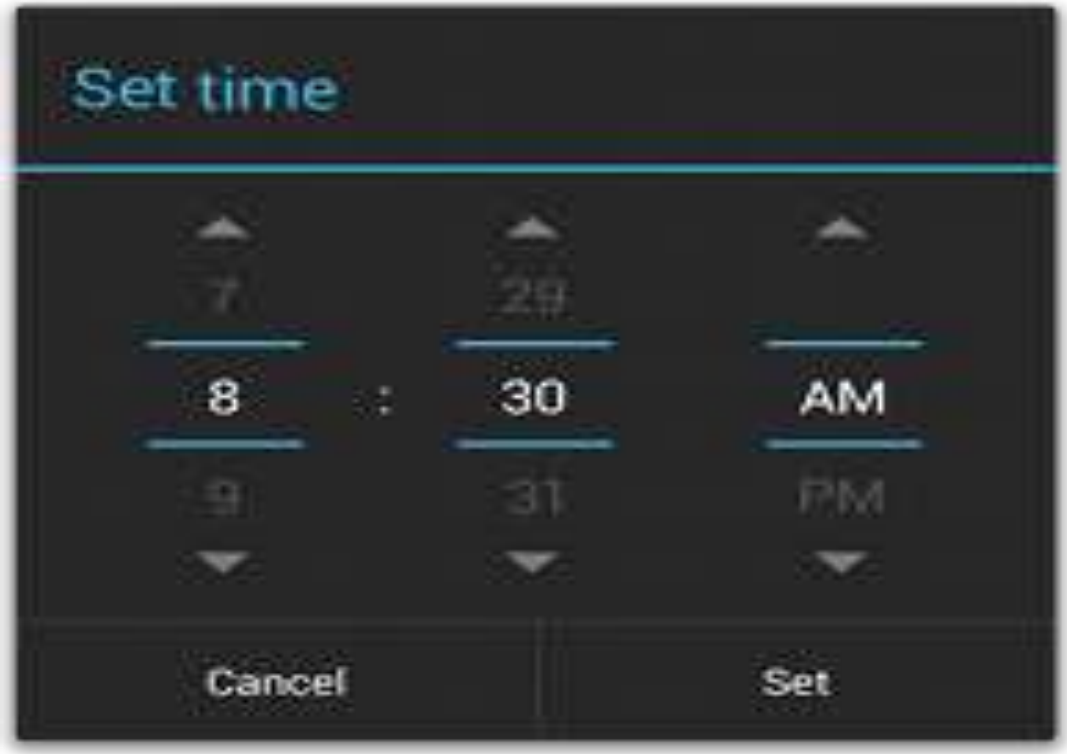
Pickers

- Android provides controls for the user to pick a time or pick a date as ready-to-use dialogs.
- Each picker provides controls for selecting each part of the time (hour, minute, AM/PM) or date (month, day, year).
- Using these pickers helps ensure that your users can pick a time or date that is valid, formatted correctly, and adjusted to the user's locale.

Pickers

- Android provides controls for the user to pick a time or pick a date as ready-to-use dialogs.
- Each picker provides controls for selecting each part of the time (hour, minute, AM/PM) or date (month, day, year).
- Using these pickers helps ensure that your users can pick a time or date that is valid, formatted correctly, and adjusted to the user's locale.

Pickers


Set time
7
8
9
29
30
31
AM
PM
Cancel
Set

Set date
Sep
Oct
Nov
06
07
08
2010
2011
2012
Cancel
Set

Input Events

- On Android, there's more than one way to intercept the events from a user's interaction with your application.
- When considering events within your user interface, the approach is to capture the events from the specific View object that the user interacts with.
- The View class provides the means to do so.

Input Events

- Within the various View classes that you'll use to compose your layout, you may notice several public callback methods that look useful for UI events.
- These methods are called by the Android framework when the respective action occurs on that object.
- For instance, when a View (such as a Button) is touched, the **onTouchEvent( )** method is called on that object.
- However, in order to intercept this, you must extend the class and override the method.

Input Events

- However, extending every View object in order to handle such an event would not be practical.
- This is why the View class also contains a collection of nested interfaces with callbacks that you can much more easily define.
- These interfaces, called event listeners, are your ticket to capturing the user interaction with your UI.

Input Events

- While you will more commonly use the event listeners to listen for user interaction, there may come a time when you do want to extend a View class, in order to build a custom component.
- Perhaps you want to extend the Button class to make something more fancy.
- In this case, you'll be able to define the default event behaviors for your class using the class event handlers.

Event Listeners

- An event listener is an interface in the View class that contains a single callback method.
- These methods will be called by the Android framework when the View to which the listener has been registered is triggered by user interaction with the item in the UI.

- Included in the event listener interfaces are the following callback methods:
  - **onClick( )**

    From View.OnClickListener.

    This is called when the user either touches the item (when in touch mode), or focuses upon the item with the navigation-keys or trackball and presses the suitable "enter" key or presses down on the trackball.
  - **onLongClick( )**

    From View.OnLongClickListener.

    This is called when the user either touches and holds the item (when in touch mode), or focuses upon the item with the navigation-keys or trackball and presses and holds the suitable "enter" key or presses and holds down on the trackball (for one second).
  - **onFocusChange( )**

    From View.OnFocusChangeListener.

    This is called when the user navigates onto or away from the item, using the navigation-keys or trackball.

- **onKey( )**

  From View.OnKeyListener.

  This is called when the user is focused on the item and presses or releases a hardware key on the device.

- **onTouch( )**

  From View.OnTouchListener.

  This is called when the user performs an action qualified as a touch event, including a press, a release, or any movement gesture on the screen (within the bounds of the item).

- **onCreateContextMenu( )**

  From View.OnCreateContextMenuListener.

  This is called when a Context Menu is being built (as the result of a sustained "long click").

The example below shows how to register an on-click listener for a Button.

```
// Create an anonymous implementation of OnClickListener
private OnClickListener mCorkyListener = new OnClickListener() {
    public void onClick(View v) {
      // do something when the button is clicked
    }
};

protected void onCreate(Bundle savedValues) {
    ...
    // Capture our button from layout
    Button button = (Button)findViewById(R.id.corky);
    // Register the onClick listener with the implementation above
    button.setOnClickListener(mCorkyListener);
    ...
}
```

thank you
danke
謝謝
ngiyabonga
teşekkür ederim
gracias
merci
obrigado
bedankt
dziękuję
go raibh maith agat
mochchakkeram
sukriya
kop khun krap
terima kasih
grazie
arigatô
dakujem
takk
tapadh leat
спасибо
Баярлалаа
рахмат
dank je
mersi
barka
welalin
tack
spas
kia ora
vinaka
спасибі
blagodaram
misaotra
matondo
paldies
grazzi
mahalo
хвала
asante
manana
obrigada
tenki
murakoze
chokrane
mamnun
дякую
djiere dieuf
tau
akun
dankon
aciú
köszönöm
mauruuru
hvala
gràçie
bayarlalaa
nanni
nandri
kiitos
dankie
dhanyavad
enkosi
sobodi
dèkuji
sagolun
mèsi
didi madloba
najis tuke
kam sah hamnida
rahmat
chnorakaloutioun
gratias ago
gràcies
sulpáy
taiku
ありがとう
tanemirt
rahmet
diolch
dhanyavadagalu
shukriya
merce
мерси
trugarez
ευχαριστώ
xièxie
감사합니다
তোমাকে ধন্যবাদ

Thank you!