

# Basic UI Operation

# Android Activity

- Interact with users
- Present a visual user interface
- The visual content of the window is provided by a hierarchy of **views**
- Activity executed in foreground
- An application might contains one or several activities
- Class Activity

```
package tw.nthu.cs241001.helloandroid;

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

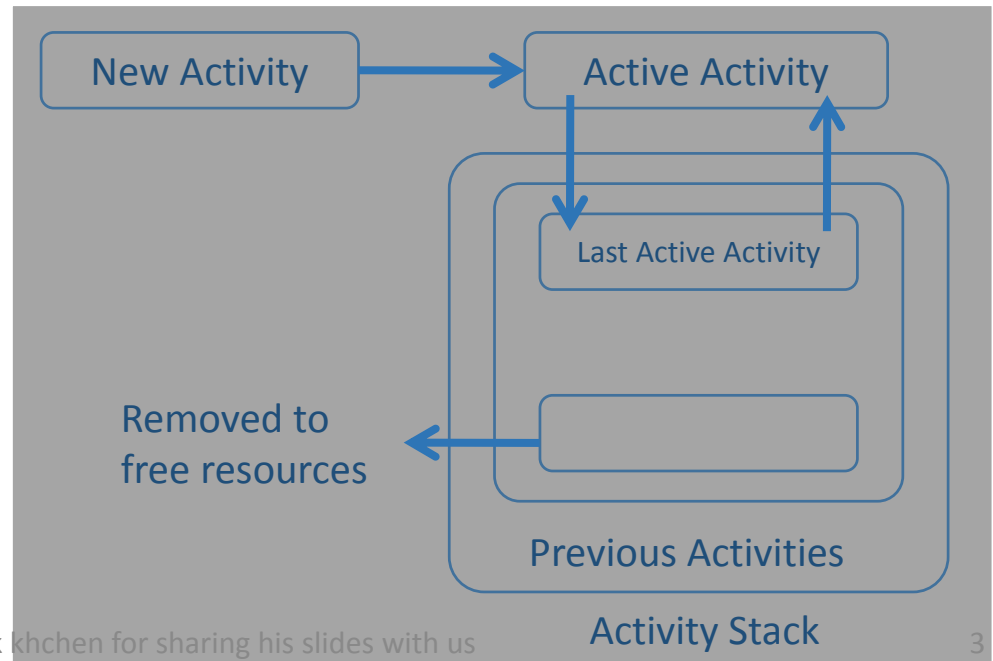
public class HelloAndroid extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

The class you create for your application to take care of creating window must extend Activity

You place your UI with [setContentView\(View\)](#)

# Activity Life Cycle

- Activities in the system are managed as an *activity stack*. When a new activity is started, it is placed on the top of the stack and becomes the running activity
- Active / running: activity in the foreground
- Pause: An activity has lost focus but is still visible
- Stopped: It's no longer visible but still retains all state and member information
- Finish / kill





# Application Components

- Activities
- Services
  - Doesn't have a visual user interface, but rather runs in the background for an indefinite period of time
- Content Provider
  - A shareable data store.
- Intents
  - A simple message-passing framework
- Broadcast Receivers
  - Receive and react to broadcast announcements
- Notifications
  - A user notification framework. Signal users without stealing focus or interrupting their current activities.
- <http://developer.android.com/guide/topics/fundamentals.html>

# Application Manifest

- Every application must have an `AndroidManifest.xml` file
- Presents essential information about the application to the Android system
- Those components using in the application should be declare in Manifest

# Application Manifest – Example

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="tw.nthu.cs241001"
  android:versionCode="1"
  android:versionName="1.0">
  <application android:icon="@drawable/icon" android:label="@string/app_name">
    <activity android:name=".IntentEx"
      android:label="@string/app_name">
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>
  </application>
</manifest>
```

The Java package for the application

Describe the application

Describe the components of the application: activity, intent

# Example 1

Button

TextView

EditText



# User Interface

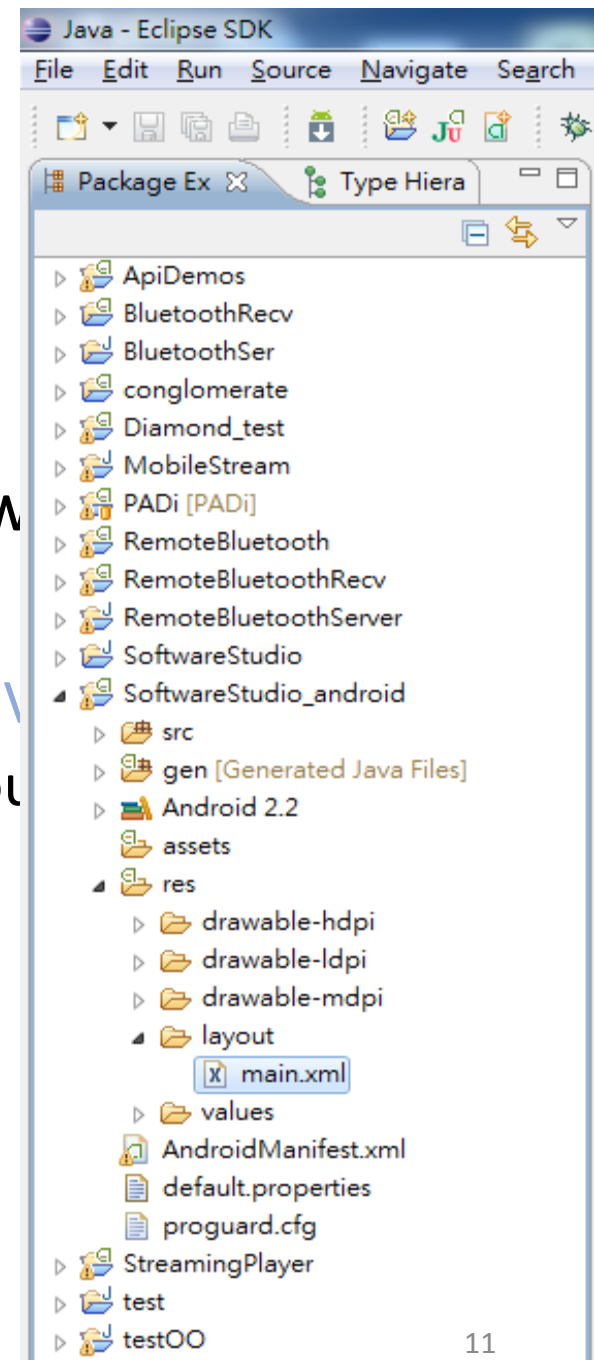
- View
  - Represents the basic building block for user interface components
  - Occupies a rectangular area on the screen and is responsible for drawing and event handling
- Widget
  - A View object that serves as an interface for interaction with the user
  - buttons, checkboxes, and text-entry fields
- Layout
  - Define the layout for your views

# Main.xml

- res → layout → main.xml
- The main layout of the application window
- Contain Layout and main.xml
  - Layout : a GUI to edit the component([View](#)) layout
  - Main.xml : the .xml format of the layout

# Main.xml

- res → layout → main.xml
- The main layout of the application will be defined in main.xml
- Contain Layout and main.xml
  - Layout : a GUI to edit the component(\res\layout)
  - Main.xml : the .xml format of the layout



# Layout

- Linear Layout(default)
  - Linearly set the View component
- Absolute Layout
  - Set the View component at where you put it
- Change layout to LinearLayout in Main.xml Layout
  - Select screen → right click → remove
  - Select Layout → LinearLayout → Drop to the screen
  - Properties → ID → Change the Layout name

# TextView

- In main.xml Layout → Select **Views (Form Widgets)** → **TextView** → drop to the screen
- Properties → Property
  - right click → edit **Id** : Name of the TextView(also seen at right side of Eclipse → Layout)  
**Do not remove keywords “@+id/”**
  - right click → edit **Text** : the showing text of the text field
  - right click → **Layout height / Layout width** : the height/width of the text feild ( unit : **px** or **dip**)
- API : `setText( String )`

Java - SoftwareStudio\_android/res/layout/main.xml - Eclipse SDK

File Edit Run Navigate Search Project Refactor Window Help

Package Ex Type Hiera

main.xml

Editing config: default

3.7in WVGA (Nexus One) Portrait Normal Day time

Palette

Form Widgets

TextView Large Medium Small Button

OFF  CheckBox  RadioButton

CheckedTextView

Text Fields

Layouts

Composite

Images & Media

Time & Date

Transitions

Advanced

Custom & Library Views

SoftwareStudio\_android

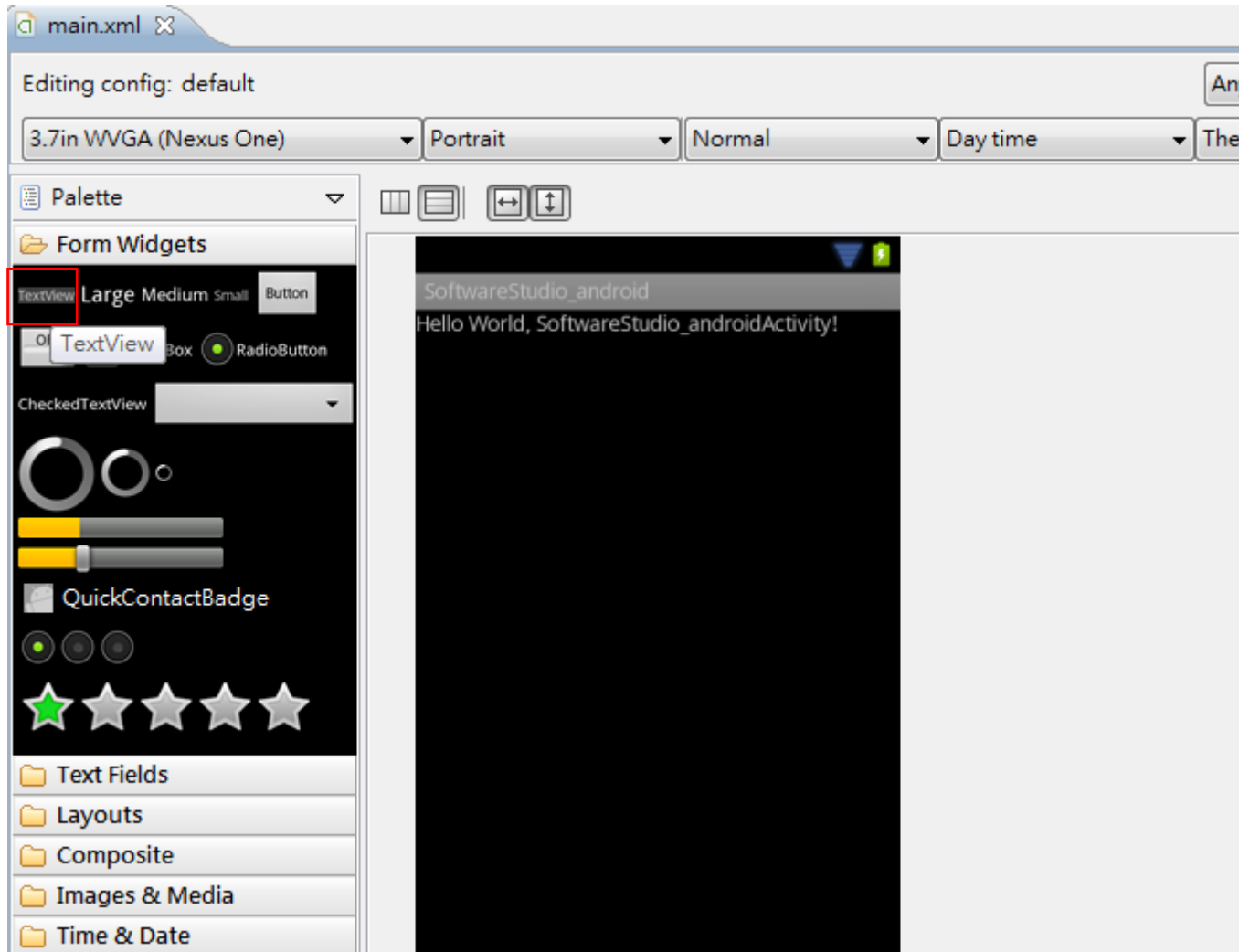
Hello World, SoftwareStudio\_androidActivity!

The project target (Android 2.3.3) is still loading.  
main.xml will refresh automatically once the process is finished.

We thank khchen for sharing his slides with us

14

- In main.xml Layout → Select **Views** → **TextView** → drop to the screen



We thank khchen for sharing his slides with us

- Properties → Property

- right click → edit **Id** :  
Name of the  
TextView(also seen at  
Eclipse → Layout)

Do not remove  
keywords “@+id/”

- right click → edit **Text** :  
the showing text of the  
text field

- right click → **Layout  
height / Layout width** :  
the height/width of  
the text feild ( unit : **px**  
or **dip**)

The screenshot shows the Android Studio interface. At the top, there are settings for the device: "3.7in WVGA (Nexus One)", "Portrait", "Normal", and "Day time". Below this is a toolbar with various icons. The main area is divided into two panes. The left pane is the "Palette" showing various "Form Widgets" such as "TextView", "Button", "CheckBox", "RadioButton", "CheckedTextView", "QuickContactBadge", "Text Fields", "Layouts", "Composite", "Images & Media", "Time & Date", "Transitions", "Advanced", and "Custom & Library Views". The right pane is the "Graphical Layout" showing a preview of the app's UI. It displays a "SoftwareStudio\_android" activity with the text "Hello World SoftwareStudio\_androidActivity!" and a "TextView" widget highlighted with a blue border. At the bottom, there is a "Properties" window showing a table of properties for the selected "TextView" widget.

Property	Value
Auto link	
Background	
Buffer type	
Clickable	
Content description	
Cursor visible	
Digits	
Drawable bottom	
Drawable left	
Drawable padding	



- Properties → Property

- TextView → Id : Name of the TextView (also seen at right side of Eclipse → Layout)

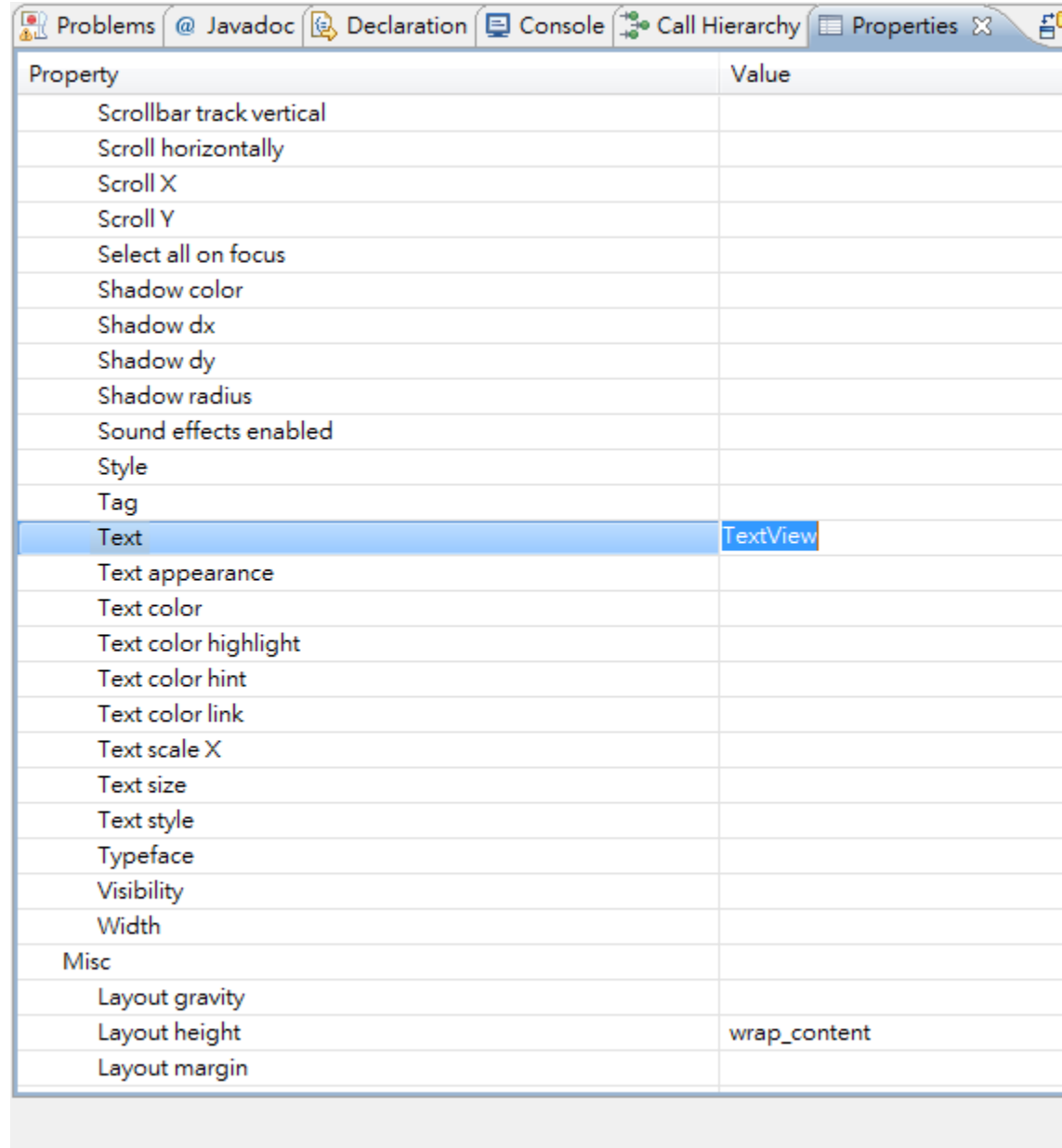
Do not remove keywords “@+id/”

The screenshot shows the Eclipse IDE's Properties window for a TextView widget. The window has tabs for Problems, Javadoc, Declaration, Console, Call Hierarchy, and Properties. The Properties window is divided into two columns: Property and Value. The 'Id' property is selected and highlighted in blue, with its value '@+id/textView1' displayed in the Value column. Other properties listed include Editor extras, Ellipsize, Ems, Fade scrollbars, Fading edge, Fading edge length, Fits system windows, Focusable, Focusable in touch mode, Freezes text, Gravity, Haptic feedback enabled, Hint, Ime action id, Ime action label, Ime options, Include font padding, Input type, Is scroll container, Keep screen on, Lines, Line spacing extra, Line spacing multiplier, Links clickable, Long clickable, and Marquee repeat limit.

Property	Value
Editor extras	
Ellipsize	
Ems	
Fade scrollbars	
Fading edge	
Fading edge length	
Fits system windows	
Focusable	
Focusable in touch mode	
Freezes text	
Gravity	
Haptic feedback enabled	
Height	
Hint	
<b>Id</b>	<b>@+id/textView1</b>
Ime action id	
Ime action label	
Ime options	
Include font padding	
Input type	
Is scroll container	
Keep screen on	
Lines	
Line spacing extra	
Line spacing multiplier	
Links clickable	
Long clickable	
Marquee repeat limit	

ow, to later retrieve it with `View.findViewById()` or `Activity.findViewById()`. [reference]

- TextView → Text :  
the showing text of  
the text field



Property	Value
Scrollbar track vertical	
Scroll horizontally	
Scroll X	
Scroll Y	
Select all on focus	
Shadow color	
Shadow dx	
Shadow dy	
Shadow radius	
Sound effects enabled	
Style	
Tag	
<b>Text</b>	<b>TextView</b>
Text appearance	
Text color	
Text color highlight	
Text color hint	
Text color link	
Text scale X	
Text size	
Text style	
Typeface	
Visibility	
Width	
Misc	
Layout gravity	
Layout height	wrap_content
Layout margin	

- Misc → Layout height / Layout width : the height/width of the text feild ( unit : px or dip)

Property	Value
Text color hint	
Text color link	
Text scale X	
Text size	
Text style	
Typeface	
Visibility	
Width	
▲ Misc	
Layout gravity	
Layout height	wrap_content
Layout margin	
Layout margin bottom	
Layout margin left	
Layout margin right	
Layout margin top	
Layout weight	
Layout width	76dp
▲ Deprecated	
Auto text	
Capitalize	
Editable	
Enabled	
Input method	
Numeric	
Password	
Phone number	
Single line	

# EditText

- A User edited Text field
- In main.xml Layout → Select **Views** → **EditText** → drop to the screen
- Properties → Property
  - EditText → **Id** : Name of the TextView(also seen at right side of Eclipse → Layout)
  - EditText → **Text** : the showing text of the text field
  - Misc → **Layout height / Layout width** : the height/width of the text feild
  - Misc → **Layout height / Layout width** : the height/width of the text feild ( unit : **px** or **dip**)
  - Misc → **Layout x / Layout y** : the position of the EditText ( unit : **px** or **dip**)
- API : `getText( String )`

# Button

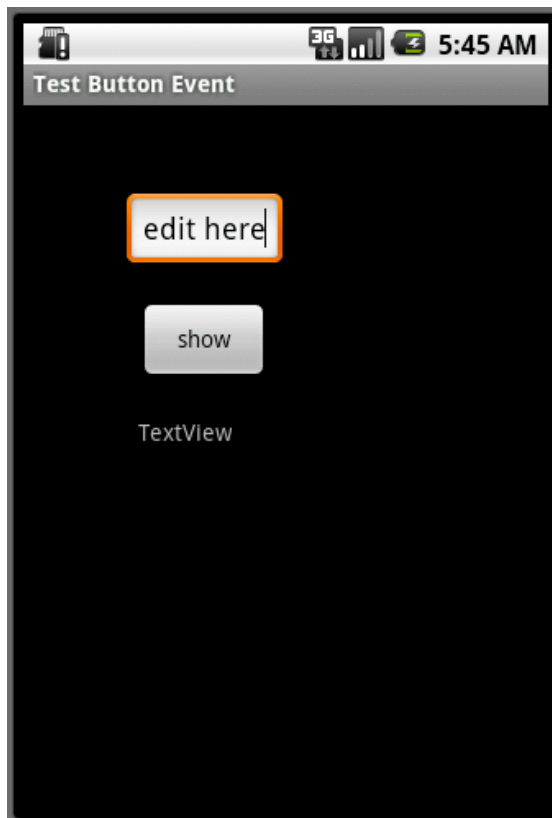
- In main.xml Layout → Select Views → Button → drop to the screen
- Properties → Property
  - Button → Id : Name of the TextView(also seen at right side of Eclipse → Layout)
  - Button → Text : the showing text of the text field
  - Misc → Layout height / Layout width : the height/width of the button
  - Misc → Layout height / Layout width : the height/width of the button ( unit : px or dip)
  - Misc → Layout x / Layout y : the position of the button( unit : px or dip)

# Event Listener

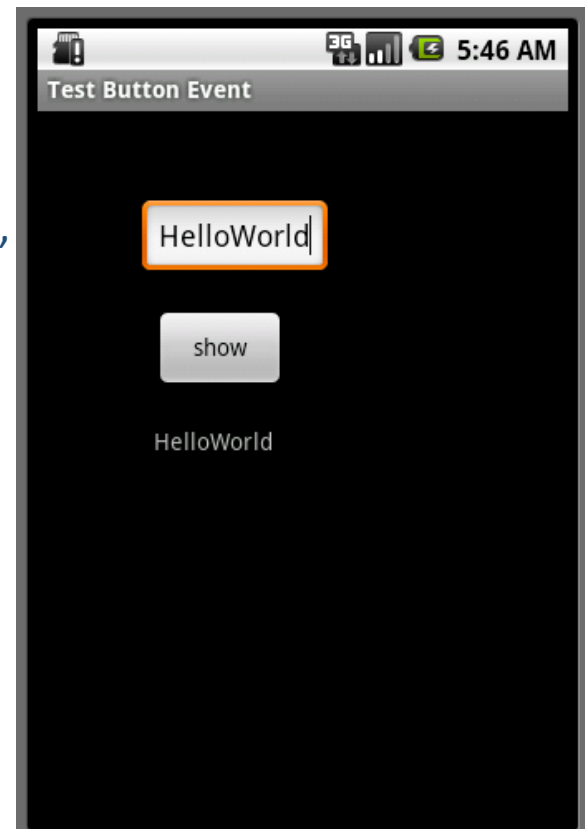
- **Event Listener** : A way for a class to provide notifications when something of interesting happens.
- Class `android.view.View`. **setOnClickListener**
  - Interface definition for a callback to be invoked when a view is clicked.
  - Public void **onClick**( View v)
    - Called when a view has been clicked

# Example – Button Click

- Click the Button and change the text of TextView



Click  
button  
show”  

# TestButtonEvent.java

```
package tw.nthu.cs241001.examples;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

public class labhello extends Activity {
    public Button myButton;
    public TextView myText;
    public EditText myEdit;
```



@Override

```
public void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.main);  
    // get Views generated by main.xml layout  
    myButton = (Button)findViewById(R.id.RButton);  
    myText = (TextView)findViewById(R.id.RText);  
    myEdit= (EditText)findViewById(R.id.REdit);  
    //set onClickListerner  
    myButton.setOnClickListener(event);  
}
```

```
private OnClickListener event = new OnClickListener(){  
    public void onClick(View v){  
        String str= myEdit.getText().toString();  
        myText.setText(str);  
    }  
};  
}
```

# Main.xml

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<AbsoluteLayout
```

```
    android:id="@+id/AbsoluteLayout01"  
    android:layout_width="fill_parent"  
    android:layout_height="fill_parent"  
    xmlns:android="http://schemas.android.com/apk/res/android">
```

```
<EditText
```

```
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:id="@+id/REdit"  
    android:layout_y="53dip"  
    android:layout_x="63dip"  
    android:text="edit here">
```

```
</EditText>
```

```
<Button
```

```
    android:id="@+id/RButton"  
    android:layout_height="50dip"  
    android:layout_width="80dip"  
    android:text="show"  
    android:layout_y="120dip"  
    android:layout_x="70dip">
```

```
</Button>
```

```
<TextView
```

```
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:id="@+id/RText"  
    android:layout_y="190dip"  
    android:layout_x="70dip"  
    android:text="TextView">
```

```
</TextView>
```

```
</AbsoluteLayout>
```

# Find API By Yourself!!

- Android API reference
  - <http://developer.android.com/reference/packages.html>
- View API
  - <http://developer.android.com/reference/android/view/View.html>

# Example2

Menu

Toast

# Menu

- A button “Menu” as a shortcut of opening menu list.
- Design menu items and its function to manage application.
- API reference

<http://developer.android.com/intl/zh-TW/reference/android/view/Menu.html>

Menu  
Shortcut  
button



# Design Menu – Data Structure

- public interface **Menu**
  - Data type of Menu component
- public interface **MenuItem**
  - Data type of Menu items



# Design Menu - API

- Create Menu
  - `public boolean onCreateOptionsMenu (Menu menu)`
- Add item
  - `public abstract MenuItem add(int groupId, int itemId, int order, CharSequence title)`
    - `groupId` : The group identifier that this item should be part of
    - `itemId` : Unique item ID
    - `order` : The order for the item
    - `Title` : The text to display for the item

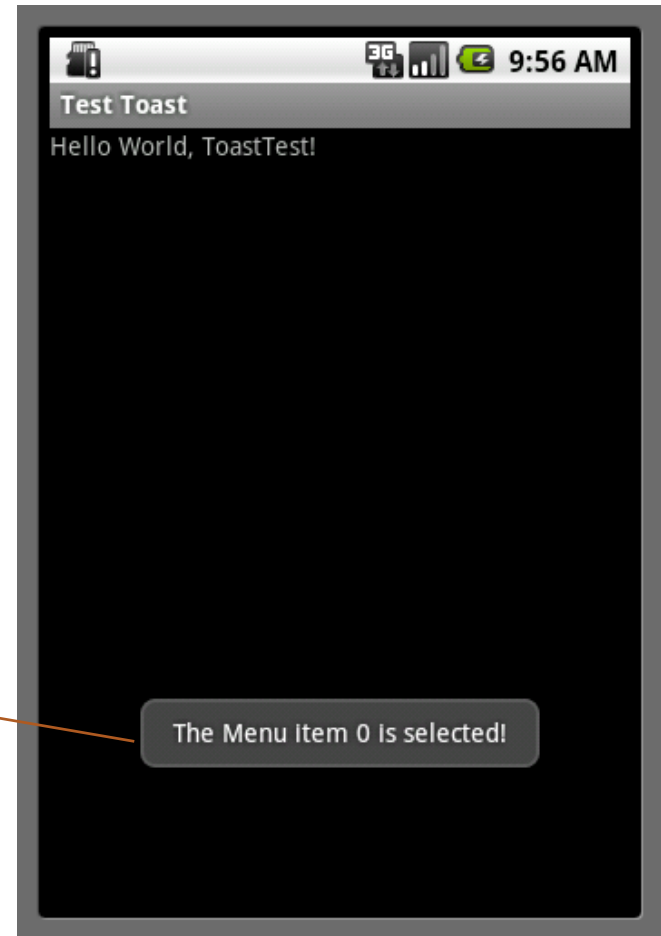
# Design Menu – API(2)

- Listen to menu item selected
  - This hook is called whenever an item in your options menu is selected
  - `public boolean onOptionsItemSelected (MenuItem item)`
- Listen to menu closed
  - `public void onOptionsItemSelected (Menu menu)`
  - This hook is called whenever the options menu is being closed (either by the user canceling the menu with the back/menu button, or when an item is selected).
- Get selected item id
  - `abstract int getItemId()`

# Toast

- Show message box on the screen
- import android.widget.Toast;

Toast Message

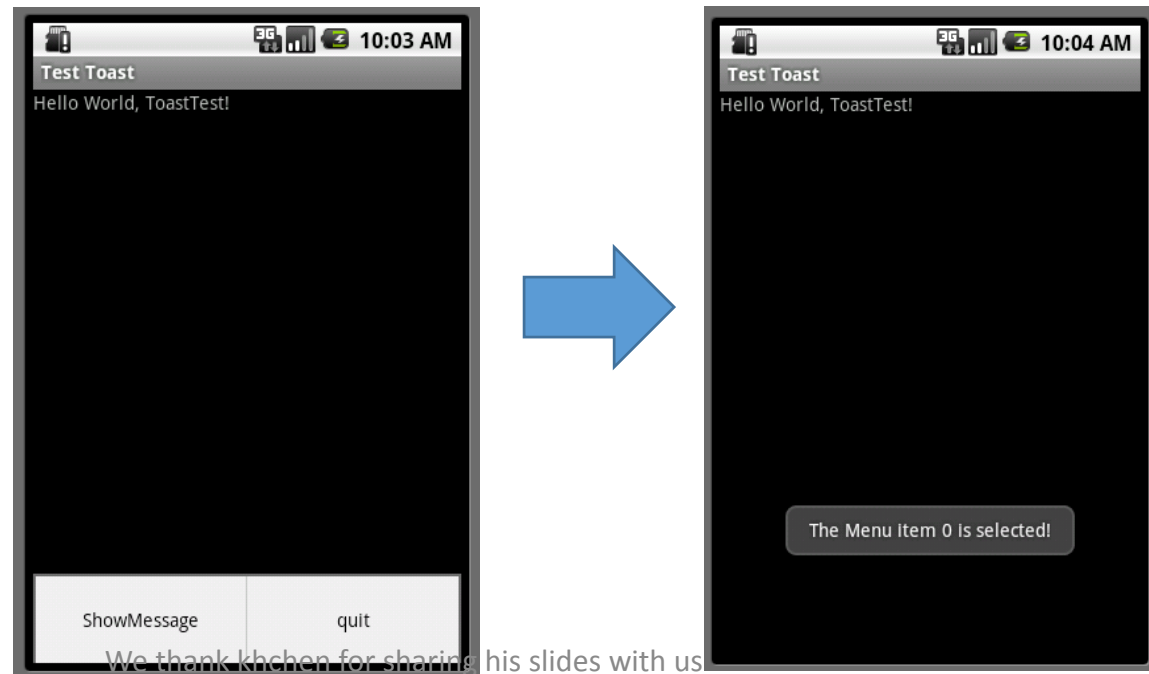


# Toast API

- public static **Toast makeText** (Context **context**, CharSequence **text**, int **duration**)
  - **context** : the context to use. Usually your **Application** or **Activity** object.
  - **text** : the text to show.
  - **duration** : How long to display the message. Either **LENGTH\_SHORT** or **LENGTH\_LONG**
    - **LENGTH\_LONG**
      - Show the view or text notification for a long period of time. This time could be user-definable
      - public void **setDuration**(int duration)
- public void **show**()
  - Show the view for the specified duration.

# Example

- Create Menu with two items
- Select ShowMessage to show toast
- Select quit to finish activity



# Example

```
package tw.nthu.cs241001.examples;

import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.Toast;

public class ToastTest extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
    public boolean onCreateOptionsMenu(Menu menu)
    {
        menu.add( 0,0,0 ,"ShowMessage");
    }
}
```

```

public boolean onOptionsItemSelected(MenuItem item)
{
    // get selected item
    super.onOptionsItemSelected(item);
    // get selected item id and do the specify function
    switch(item.getItemId())
    {
        case 0:
            // show toast message
            Toast.makeText(
                this,
                "The Menu item 0 is selected!",
                Toast.LENGTH_LONG).show();

            break;

        case 1:
            // finish ap at quit select
            finish();

            break;
    }
    return true;
}
} // end class ToastTest

```

# Android Lab2



# Lab2

- BMI counting
  - User input height and weight, counting the BMI value and show the comment.
  - Press “BMI” button to show the counting result.
    - $BMI = \text{weight}(\text{kg}) / \text{height}^2(\text{m}^2)$
  - Press “Comment” button to show the comment
    - $BMI < 18$  : too light
    - $18 \leq BMI < 24$  : normal
    - $BMI \geq 24$  : too heavy

# Lab Requirement

- Basic Requirement
  - Two Button : BMI , Comment
  - Two EditText : User enter field for height and weight
  - Two TextView : One to show the BMI value and the other show the comment

# Lab Requirement

- Layout Example

Height(m) 1.6

Weight(kg) 50

BMI 19

Comment Normal



EditText



Button



TextView

# Useful Method

- *String* *CharSequence.toString()*
  - Parsing CharSwquence to String
  - Eg. `TextView.getText().toString`
- *int* *Integer.parseInt( string str )*
  - Parsing string to integer
- *String* *Integer.toString( Interger integer )*
  - Parsing ineteger to string

# Useful Method

- *String CharSequence.toString()*
  - Parsing CharSwquence to String
  - Eg. `TextView.getText().toString`
- *float Float.parseFloat( string str )*
  - Parsing string to float
- *String Float.toString(Float fp )*
  - Parsing float to string