User Interface

Introduction

- Android provides a simple framework with “out of the box” controls we can use.
- All available controls are based on two parent classes: `View` and `ViewGroup`. `ViewGroup` extends `View`.

Introduction

- The android platform enables developing the GUI in several different approaches.
- One approach is based on coding everything in our source code.
- Another approach is based on defining the user interface in XML document.
- A third approach includes both defining the user interface in XML document and referring it from within the source code.

GUI (in Source Code) Sample

```
package com.abelski.android;

import android.app.Activity;
import android.os.Bundle;
import android.view.ViewGroup.LayoutParams;
import android.widget.LinearLayout;
import android.widget.TextView;

public class SimpleGUIInCode extends Activity
{
    private LinearLayout firstNameContainer;
    private LinearLayout lastNameContainer;
    private LinearLayout emailContainer;
    private LinearLayout addressContainer;
    private LinearLayout parentContainer;

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

GUI (in Source Code) Sample

```
    createEmailContainer();
    createAddressContainer();
    createParentContainer();
    setContentView(parentContainer);
}

private void createFirstNameContainer()
{
    firstNameContainer = new LinearLayout(this);
    firstNameContainer.setLayoutParams(new LayoutParams(
            LayoutParams.FILL_PARENT, LayoutParams.WRAP_CONTENT));
    firstNameContainer.setOrientation(LinearLayout.HORIZONTAL);
    TextView firstNameLabel = new TextView(this);
    firstNameLabel.setText("First Name: ");
    firstNameContainer.addView(firstNameLabel);
    TextView firstNameValue = new TextView(this);
    firstNameValue.setText("Haim");
    firstNameContainer.addView(firstNameValue);
}
```

GUI (in Source Code) Sample

```
private void createLastNameContainer()
{
    lastNameContainer = new LinearLayout(this);
    lastNameContainer.setLayoutParams(new LayoutParams(
            LayoutParams.FILL_PARENT, LayoutParams.WRAP_CONTENT));
    lastNameContainer.setOrientation(LinearLayout.HORIZONTAL);
    TextView lastNameLabel = new TextView(this);
    lastNameLabel.setText("Last Name: ");
    lastNameContainer.addView(lastNameLabel);
    TextView lastNameValue = new TextView(this);
    lastNameValue.setText("Michael");
    lastNameContainer.addView(lastNameValue);
}

private void createEmailContainer()
{
    emailContainer = new LinearLayout(this);
    emailContainer.setLayoutParams(new LayoutParams(
            LayoutParams.FILL_PARENT, LayoutParams.WRAP_CONTENT));
    emailContainer.setOrientation(LinearLayout.HORIZONTAL);
    TextView emailLabel = new TextView(this);
    emailLabel.setText("Email Address:");
```

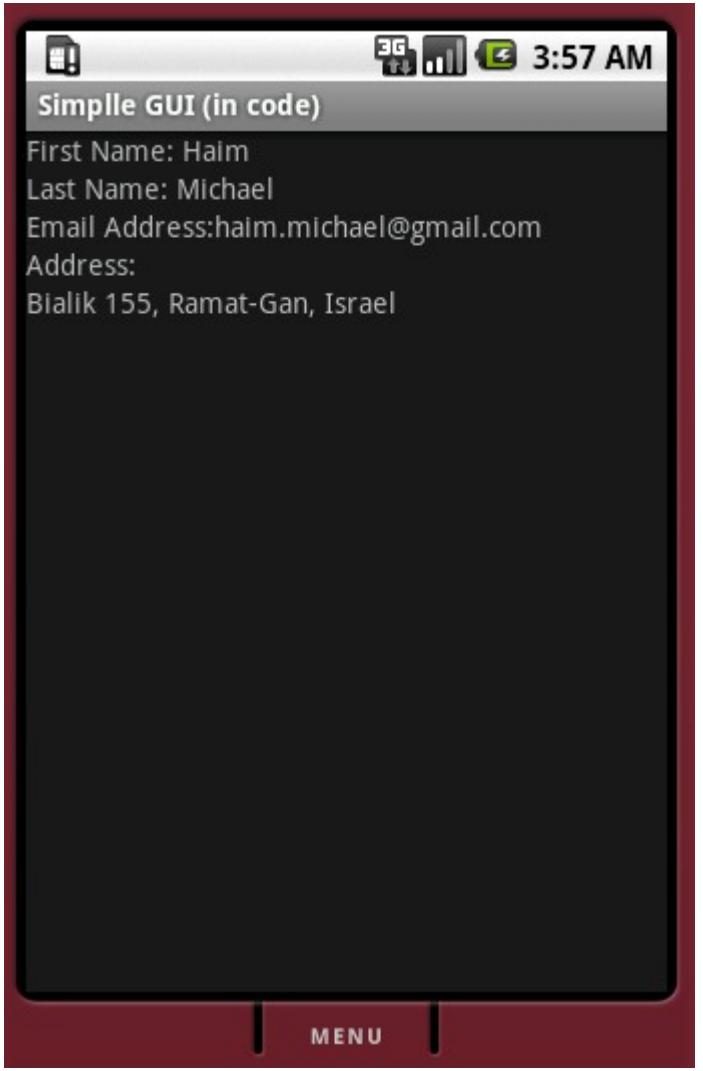
GUI (in Source Code) Sample

```
    TextView emailValue = new TextView(this);
    emailValue.setText("haim.michael@gmail.com");
    emailContainer.addView(emailLabel);
    emailContainer.addView(emailValue);
}

private void createAddressContainer()
{
    addressContainer = new LinearLayout(this);
    addressContainer.setLayoutParams(new LayoutParams(
            LayoutParams.FILL_PARENT, LayoutParams.WRAP_CONTENT));
    addressContainer.setOrientation(LinearLayout.VERTICAL);
    TextView addressLabel = new TextView(this);
    addressLabel.setText("Address:");
    TextView addressValue = new TextView(this);
    addressValue.setText("Bialik 155, Ramat-Gan, Israel");
    addressContainer.addView(addressLabel);
    addressContainer.addView(addressValue);
}
```

GUI (in Source Code) Sample

```
    private void createParentContainer()
    {
        parentContainer = new LinearLayout(this);
        parentContainer.setLayoutParams(new LayoutParams(
                LayoutParams.FILL_PARENT, LayoutParams.FILL_PARENT));
        parentContainer.setOrientation(LinearLayout.VERTICAL);
        parentContainer.addView(firstNameContainer);
        parentContainer.addView(lastNameContainer);
        parentContainer.addView(emailContainer);
        parentContainer.addView(addressContainer);
    }
}
```

GUI (in Source Code) Sample



GUI (in XML) Sample

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

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

    <!-- FIRST NAME CONTAINER -->
    <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation="horizontal" android:layout_width="fill_parent"
        android:layout_height="wrap_content">
        <TextView android:layout_width="wrap_content"
            android:layout_height="wrap_content" android:text="First Name:" />
        <TextView android:layout_width="wrap_content"
            android:layout_height="wrap_content" android:text="Haim" />
    </LinearLayout>
```

GUI (in XML) Sample

```
<!-- LAST NAME CONTAINER -->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal" android:layout_width="fill_parent"
    android:layout_height="wrap_content">
    <TextView android:layout_width="wrap_content"
        android:layout_height="wrap_content" android:text="Last Name:" />
    <TextView android:layout_width="wrap_content"
        android:layout_height="wrap_content" android:text="Michael" />
</LinearLayout>

<!-- EMAIL ADDRESS CONTAINER -->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal" android:layout_width="fill_parent"
    android:layout_height="wrap_content">
    <TextView android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Email Address:" />
    <TextView android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Haim.Michael@gmail.com " />
</LinearLayout>
```

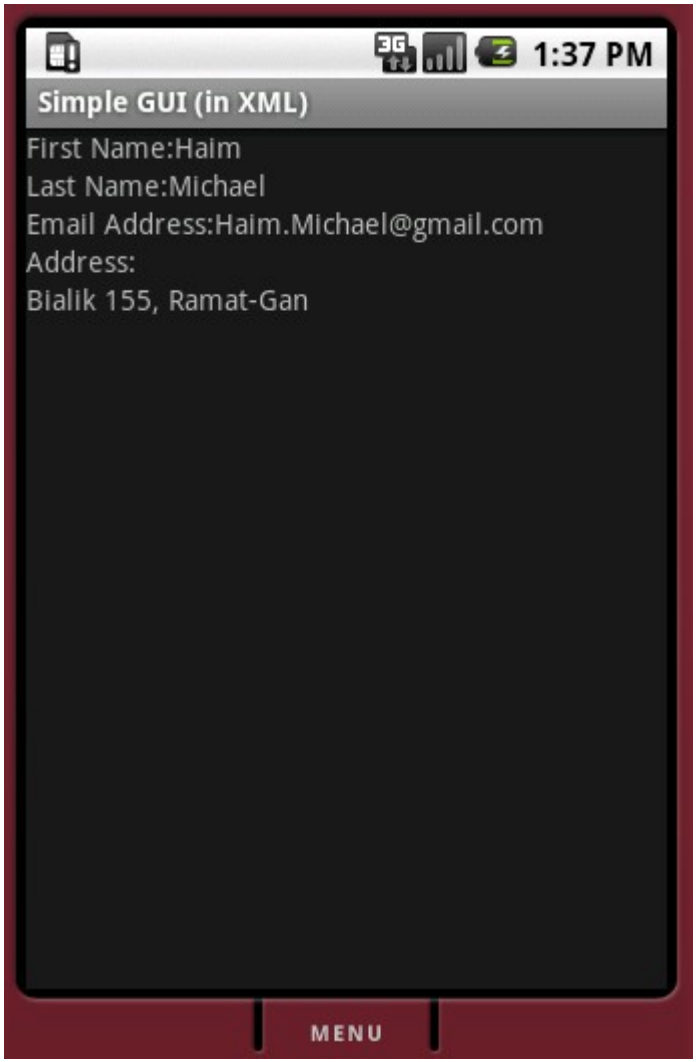
GUI (in XML) Sample

```
    <!-- ADDRESS CONTAINER -->
    <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation="vertical" android:layout_width="fill_parent"
        android:layout_height="wrap_content">
        <TextView android:layout_width="fill_parent"
            android:layout_height="wrap_content" android:text="Address: " />
        <TextView android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Bialik 155, Ramat-Gan" />
    </LinearLayout>

</LinearLayout>
```

GUI (in XML) Sample

```
package com.abelski.android;

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

public class SimpleGUIInXML extends Activity {

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

GUI (in XML) Sample



GUI (in XML & Code) Sample

```
<?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">

    <!-- FIRST NAME CONTAINER -->
    <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation="horizontal" android:layout_width="fill_parent"
        android:layout_height="wrap_content">
        <TextView android:id="@+id/firstNameTextLabel"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"/>
        <TextView android:id="@+id/firstNameTextValue"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />
    </LinearLayout>

<!-- LAST NAME CONTAINER -->
    <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation="horizontal" android:layout_width="fill_parent"
        android:layout_height="wrap_content">
```

GUI (in XML & Code) Sample

```
    <TextView android:id="@+id/lastNameTextLabel"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>
    <TextView android:id="@+id/lastNameTextValue"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
</LinearLayout>

<!-- EMAIL CONTAINER -->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal" android:layout_width="fill_parent"
    android:layout_height="wrap_content">
    <TextView android:id="@+id/emailTextLabel"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>
    <TextView android:id="@+id/emailTextValue"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
</LinearLayout>
```

GUI (in XML & Code) Sample

```
    <!-- ADDRESS CONTAINER -->
    <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation="vertical" android:layout_width="fill_parent"
        android:layout_height="wrap_content">
        <TextView android:id="@+id/addressLabelText"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content" />
        <TextView android:id="@+id/addressValueText"
            android:layout_height="wrap_content"
            android:layout_width="fill_parent" />
    </LinearLayout>

</LinearLayout>
```

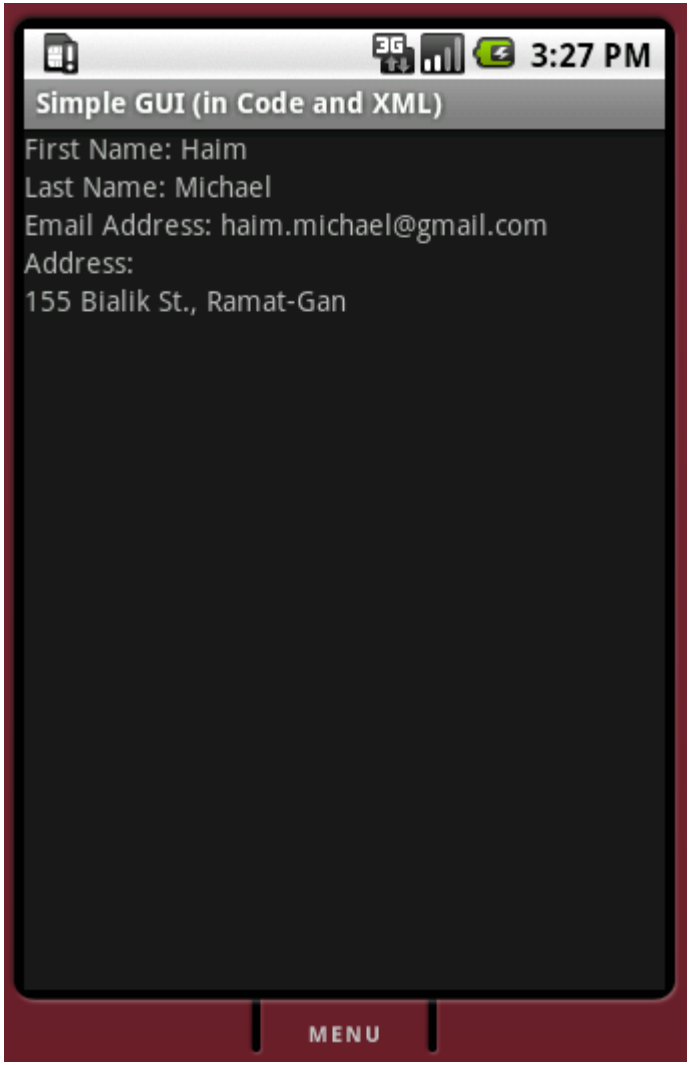
GUI (in XML & Code) Sample

```
package com.abelski.android;

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

public class SimpleGUIinCodeXML extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        TextView firstNameLabel = (TextView)findViewById(R.id.firstNameTextLabel);
        firstNameLabel.setText("First Name: ");
        TextView firstNameValue = (TextView)findViewById(R.id.firstNameTextValue);
        firstNameValue.setText("Haim");
        TextView lastNameLabel = (TextView)findViewById(R.id.lastNameTextLabel);
```

GUI (in XML & Code) Sample

```
        lastNameLabel.setText("Last Name: ");
        TextView lastNameValue = (TextView)findViewById(R.id.lastNameTextValue);
        lastNameValue.setText("Michael");
        TextView emailLabel = (TextView)findViewById(R.id.emailTextLabel);
        emailLabel.setText("Email Address: ");
        TextView emailValue = (TextView)findViewById(R.id.emailTextValue);
        emailValue.setText("haim.michael@gmail.com");
        TextView addressLabel = (TextView)findViewById(R.id.addressLabelText);
        addressLabel.setText("Address: ");
        TextView addressValue = (TextView)findViewById(R.id.addressValueText);
        addressValue.setText("155 Bialik St., Ramat-Gan");
    }
}
```

GUI (in XML & Code) Sample



TextView

- Present text without allowing the user to edit it.
- Using the `Linkify` class we can cause the android platform to automatically present any URL address, any email address and any telephone number as one the user can easily clicks on and navigates to.

```
TextView addressValue =
      (TextView)findViewById(R.id.addressValueText);
addressValue.setText("The website is www.zindell.com, the email
      address is info@zindell.com and the telephone number
      is +972-54-6655837.");
Linkify.addLinks(addressValue, Linkify.ALL);
```

TextView

```
package com.abelski.android;

import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;
import android.text.util.Linkify;

public class SimpleTextViewSample extends Activity
{
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        TextView addressValue =
            (TextView)findViewById(R.id.addressValueText);
        addressValue.setText("The website is www.zindell.com, the email
            address is info@zindell.com and the telephone number is +972-
            54-6655837.");
        Linkify.addLinks(addressValue, Linkify.ALL);
    }
}
```

TextView

```
<?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">

    <LinearLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation="vertical" android:layout_width="fill_parent"
        android:layout_height="wrap_content">
        <TextView android:id="@+id/addressValueText"
            android:layout_height="wrap_content"
            android:layout_width="fill_parent" />
    </LinearLayout>

</LinearLayout>
```

TextView

TextView

TextView

- When using the strings as resources (defined within the `strings.xml` file) we can add markups to specify the required style.

TextView

```
package com.abelski.android;

import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;
import android.text.util.Linkify;

public class SimpleTextViewSample extends Activity
{
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        TextView addressValue =
            (TextView)findViewById(R.id.addressValueText);
        addressValue.setText(R.string.address_str);
        Linkify.addLinks(addressValue, Linkify.ALL);
    }
}
```

TextView

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello World, SimpleGUIinCodeXML!</string>
    <string name="app_name">Simple GUI (in Code and XML)</string>
    <string name="address_str">The <b>website</b> is www.zindell.com, the
        <i>email</i> address is info@zindell.com and the telephone number is
        +972-54-6655837.</string>
</resources>
```

TextView

```
<?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">

    <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation="vertical" android:layout_width="fill_parent"
        android:layout_height="wrap_content">
        <TextView android:id="@+id/addressValueText"
            android:layout_height="wrap_content"
            android:layout_width="fill_parent" />
    </LinearLayout>

</LinearLayout>
```

TextView Style Sample

EditText

- The `EditText` class extends `TextView`. Unlike `TextView`, the `EditText` control allows the user to edit the text.
- The `EditText` control has various properties that configures its functionality.

  Using the `autoText` property we can cause it to automatically fixes spelling mistakes. Using the `capitalize` property we can makes it capitalizing first sentence letters. Using the `phoneNumber` property we can get a control that accepts phone numbers only. Using the `singleLine` property the user will be forced to enter a one line input.

EditText

- We can change the style of specific portions of the text by applying static markups directly to the strings (within the `strings.xml` resource file) the same way we can do it with TextView objects.
- We can alternatively apply the required style by using the `Spannable` class.

EditText

```
package com.abelski.android;

import android.app.Activity;
import android.graphics.Color;
import android.os.Bundle;
import android.widget.EditText;
import android.text.Spannable;
import android.text.style.BackgroundColorSpan;

public class SimpleGUISample extends Activity
{
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        EditText addressValue = (EditText)
                findViewById(R.id.addressValueText);
        addressValue.setText(R.string.address_str);
        Spannable span = addressValue.getEditableText();
        span.setSpan(new BackgroundColorSpan(Color.YELLOW), 0, 10, 0);
    }
}
```

EditText

```
<?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">

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

        <EditText android:id="@+id/addressValueText"
            android:layout_height="wrap_content"
            android:layout_width="fill_parent" />

    </LinearLayout>

</LinearLayout>
```

EditText

AutoCompleteTextView

- The `AutoCompleteTextView` class is a `TextView` with the autocomplete functionality.
- The possible texts are set using an `ArrayAdapter` object.

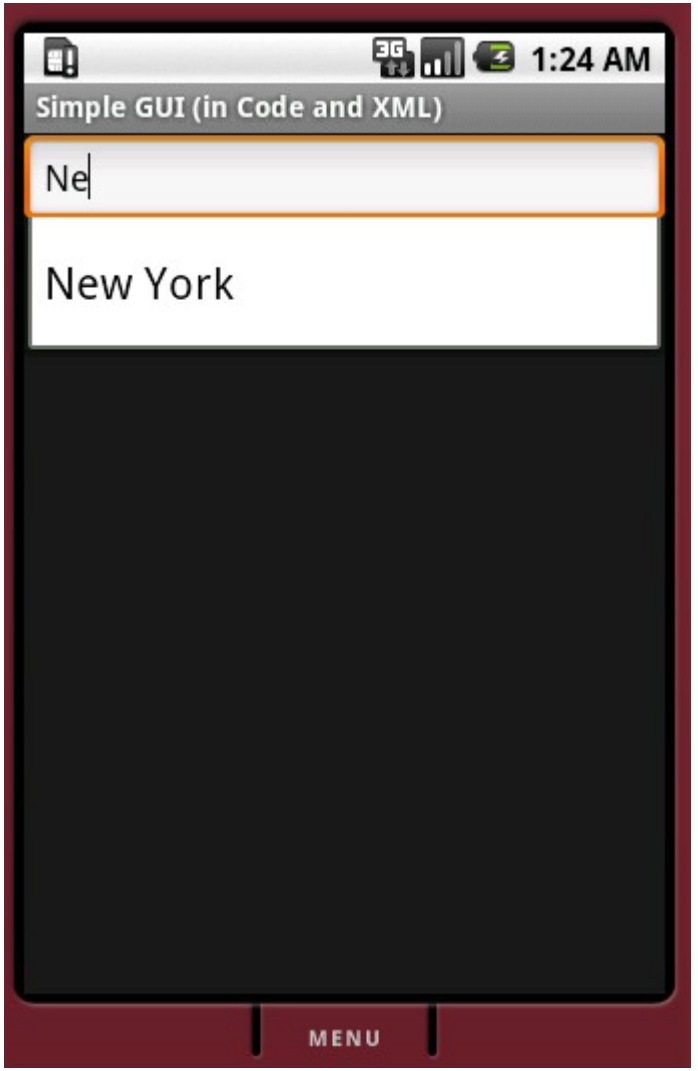
AutoCompleteTextView

```
package com.abelski.android;

import android.app.Activity;
import android.os.Bundle;
import android.widget.ArrayAdapter;
import android.widget.AutoCompleteTextView;

public class SimpleGUISample extends Activity
{

    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        AutoCompleteTextView city =
            (AutoCompleteTextView) findViewById(R.id.city);
        ArrayAdapter<String> adapter =
            new ArrayAdapter<String>(this,
            android.R.layout.simple_dropdown_item_1line,
            new String[] {"New York", "Tel Aviv", "Moscow", "Jerusalem",
            "Beirut", "London" });
        city.setAdapter(adapter);
    }
}
```

AutoCompleteTextView

```
<?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">

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

    <AutoCompleteTextView android:id="@+id/city"
        android:layout_height="wrap_content"
        android:layout_width="fill_parent" />

</LinearLayout>

</LinearLayout>
```

AutoCompleteTextView



MultiAutoCompleteTextView

- The `MultiAutoCompleteTextView` is an improvement to the `AutoCompleteTextView`.
- It allows us having the auto complete feature for more than one part of the text view.

MultiAutoCompleteTextView

```
<?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">

    <LinearLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation="vertical" android:layout_width="fill_parent"
        android:layout_height="wrap_content">
        <MultiAutoCompleteTextView android:id="@+id/colors"
            android:layout_height="wrap_content"
            android:layout_width="fill_parent" />
    </LinearLayout>

</LinearLayout>
```

MultiAutoCompleteTextView

Button

- The `Button` control is the simplest available button on the Android platform.
- Registering an events handler is done by calling the `setOnClickListener` function on the `Button` object and passing an `OnClickListener` object to it.

Button

```
package com.abelski.android;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.MultiAutoCompleteTextView;
import android.widget.TextView;


public class SimpleGUISample extends Activity
{
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button btPlus = (Button)this.findViewById(R.id.btPlus);
```

Button

```
        btPlus.setOnClickListener(new OnClickListener()
        {
            public void onClick(View v)
            {
                double firstNum =
                    Double.parseDouble(
                        ((EditText)SimpleGUISample.this.
                        findViewById(R.id.num_1))
                        .getText().toString());
                double secondNum =  Double.parseDouble(
                        ((EditText)SimpleGUISample.this.
                        findViewById(R.id.num_2)).getText().toString());
                double sum = firstNum + secondNum;
                TextView total = ((TextView)SimpleGUISample.
                        this.findViewById(R.id.sum));
                total.setText(String.valueOf(sum));
            }
        });
    }
}
```

Button

```
<?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="wrap_content">

    <EditText android:id="@+id/num_1" android:inputType="numberDecimal"
        android:layout_height="wrap_content"
        android:layout_width="fill_parent" />
    <EditText android:id="@+id/num_2" android:inputType="numberDecimal"
        android:layout_height="wrap_content"
        android:layout_width="fill_parent" />
    <Button android:id="@+id/btPlus" android:text="+"
        android:layout_height="wrap_content"
        android:layout_width="fill_parent" />
    <EditText android:id="@+id/sum" android:inputType="numberDecimal"
        android:editable="false"
        android:layout_height="wrap_content"
        android:layout_width="fill_parent" />

</LinearLayout>
```

Button

ImageButton

- The `ImageButton` control is similar to `Button`. The image can be set either dynamically by calling `setImageResource` method on the `ImageButton` object or by specifying the image in the XML layout file.
- Handling the events is as with a simple `Button`.

ImageButton

```
package com.abelski.android;

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


public class SimpleGUISample extends Activity
{
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        ImageButton btPlus = (ImageButton)this.findViewById(R.id.btPlus);
        //btPlus.setImageResource(R.drawable.plus);
        btPlus.setOnClickListener(new OnClickListener()
        {
```

ImageButton

```
            public void onClick(View v)
            {
                double firstNum =
                    Double.parseDouble(((EditText)SimpleGUISample.
                    this.findViewById(R.id.num_1)).getText().toString());
                double secondNum =
                    Double.parseDouble(((EditText)SimpleGUISample.
                    this.findViewById(R.id.num_2)).getText().toString());

                double sum = firstNum + secondNum;

                TextView total = ((TextView)SimpleGUISample.
                    this.findViewById(R.id.sum));

                total.setText(String.valueOf(sum));
            }
        });
    }
}
```

ImageButton

```
<?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="wrap_content">

    <EditText android:id="@+id/num_1" android:inputType="numberDecimal"
        android:layout_height="wrap_content"
        android:layout_width="fill_parent" />
    <EditText android:id="@+id/num_2" android:inputType="numberDecimal"
        android:layout_height="wrap_content"
        android:layout_width="fill_parent" />
    <ImageButton android:id="@+id/btPlus" android:src="@drawable/plus"
        android:layout_height="wrap_content"
        android:layout_width="fill_parent" />
    <EditText android:id="@+id/sum" android:inputType="numberDecimal"
        android:editable="false"
        android:layout_height="wrap_content"
        android:layout_width="fill_parent" />

</LinearLayout>
```

ImageButton

ToggleButton

- The `ToggleButton` control is kind of a check box. It is a two state button ('on' and 'off').
- Default behavior sets the text on this type of button either 'On' or 'Off'.
- It is possible to change the texts (instead of 'On' and 'Off') using the 'android:textOn' and 'android:textOff' properties.

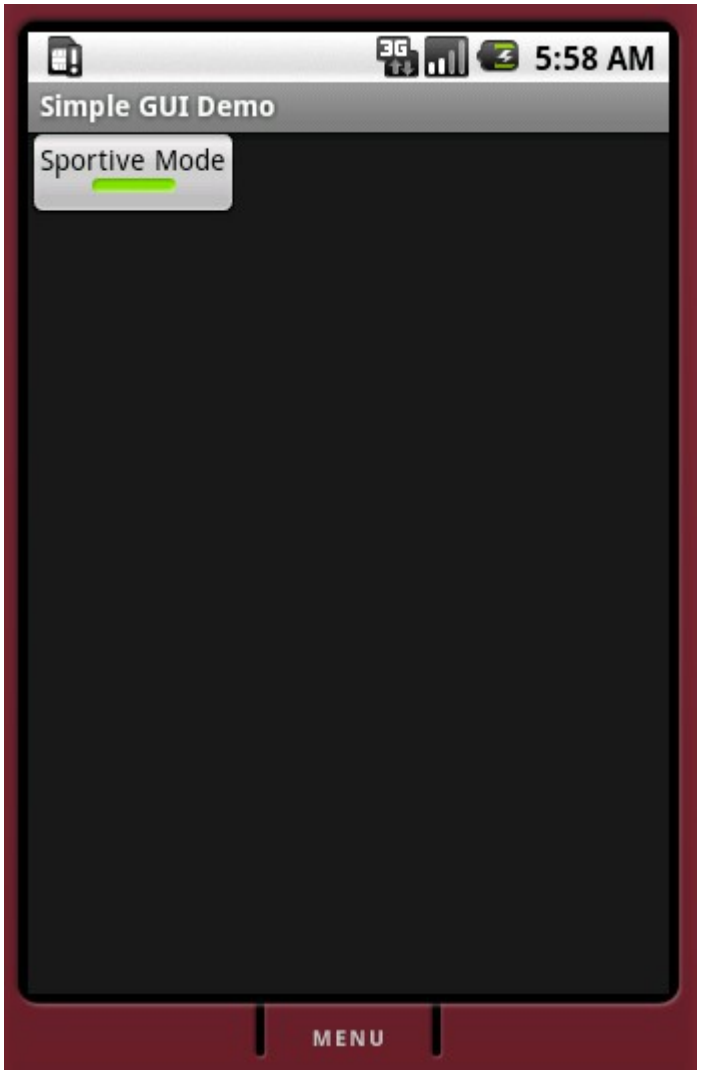
ToggleButton

```
<?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="wrap_content">

<ToggleButton android:id="@+id/bt"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:textOn="Sportive Mode"
android:textOff="Economic Mode"
android:text="Automatic Gear"/>

</LinearLayout>
```

ToggleButton

```
package com.abelski.android;

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

public class SimpleGUISample extends Activity
{
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

ToggleButton



CheckBox

- The `CheckBox` control is a simple check box similar to those we can find in other software development platforms.
- It is a two state button that allows the user to toggle its state.

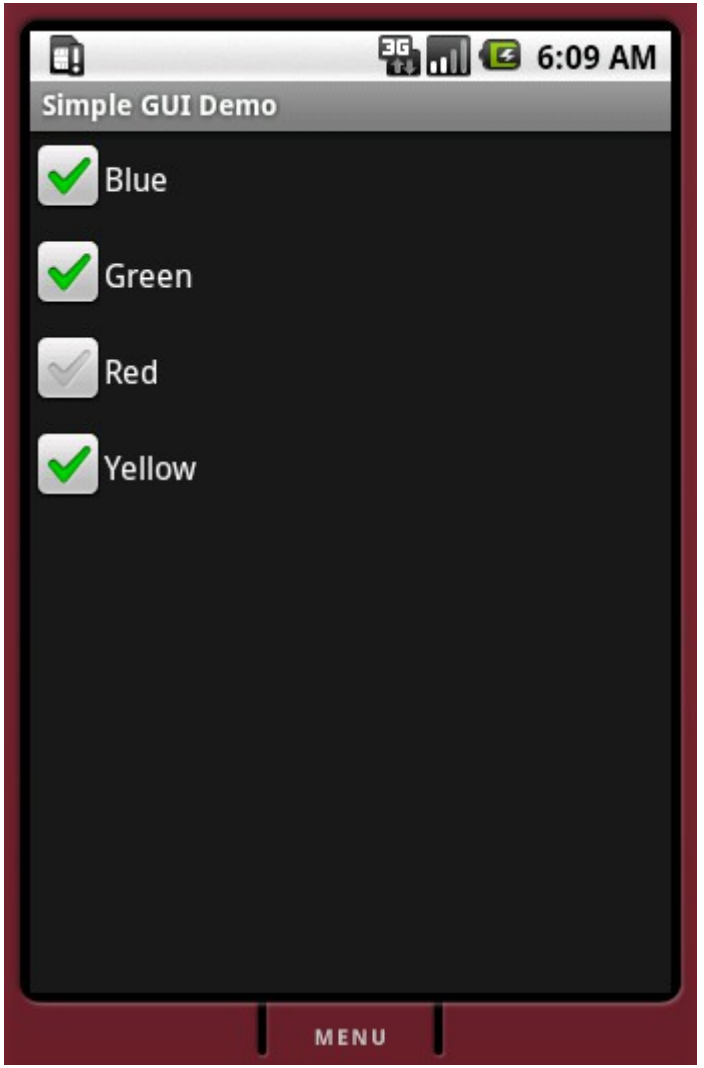
CheckBox

```
<?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="wrap_content">

<CheckBox android:text="Blue" android:layout_width="wrap_content"
android:layout_height="wrap_content" />
<CheckBox android:text="Green" android:layout_width="wrap_content"
android:layout_height="wrap_content" />
<CheckBox android:text="Red" android:layout_width="wrap_content"
android:layout_height="wrap_content" />
<CheckBox android:text="Yellow" android:layout_width="wrap_content"
android:layout_height="wrap_content" />

</LinearLayout>
```

CheckBox

```
package com.abelski.android;

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

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

CheckBox



RadioButton

- The `RadioButton` control allows selecting one option only.
- In order to allow selecting only one of the options we should place them within the same group.

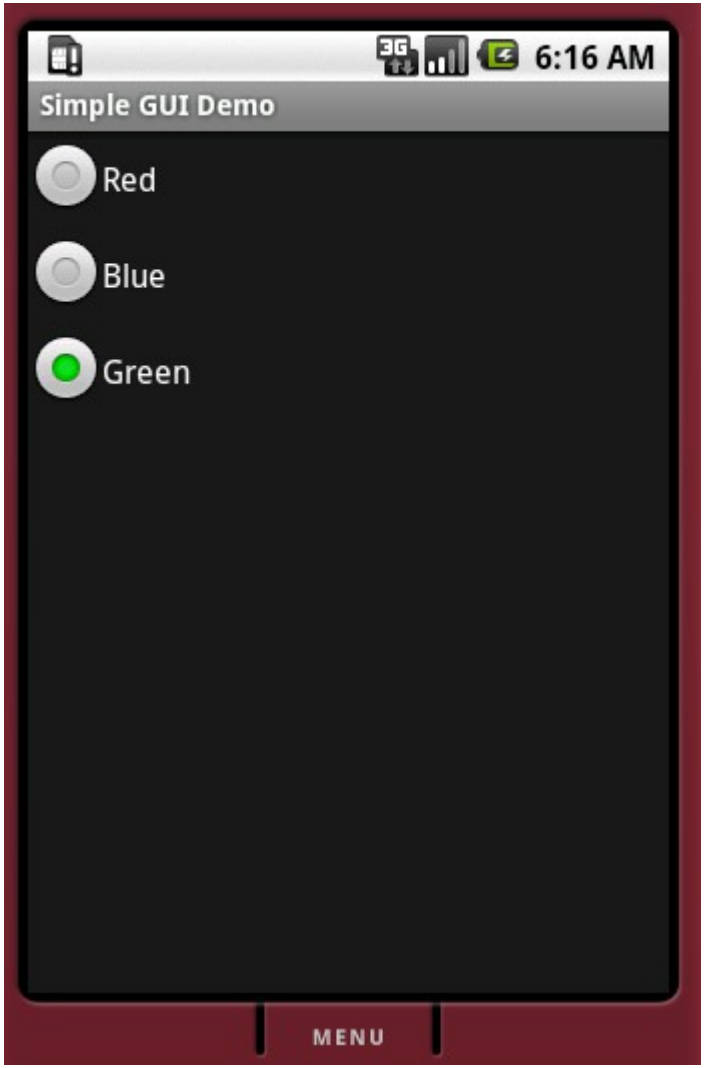
RadioButton

```
<?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="wrap_content">

<RadioGroup android:layout_width="wrap_content"
android:layout_height="wrap_content">

    <RadioButton android:id="@+id/redRBtn" android:text="Red"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <RadioButton android:id="@+id/blueRBtn" android:text="Blue"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <RadioButton android:id="@+id/greenRBtn" android:text="Green"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

</RadioGroup>

</LinearLayout>
```

RadioButton

```
package com.abelski.android;

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

public class SimpleGUISample extends Activity
{
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

RadioButton



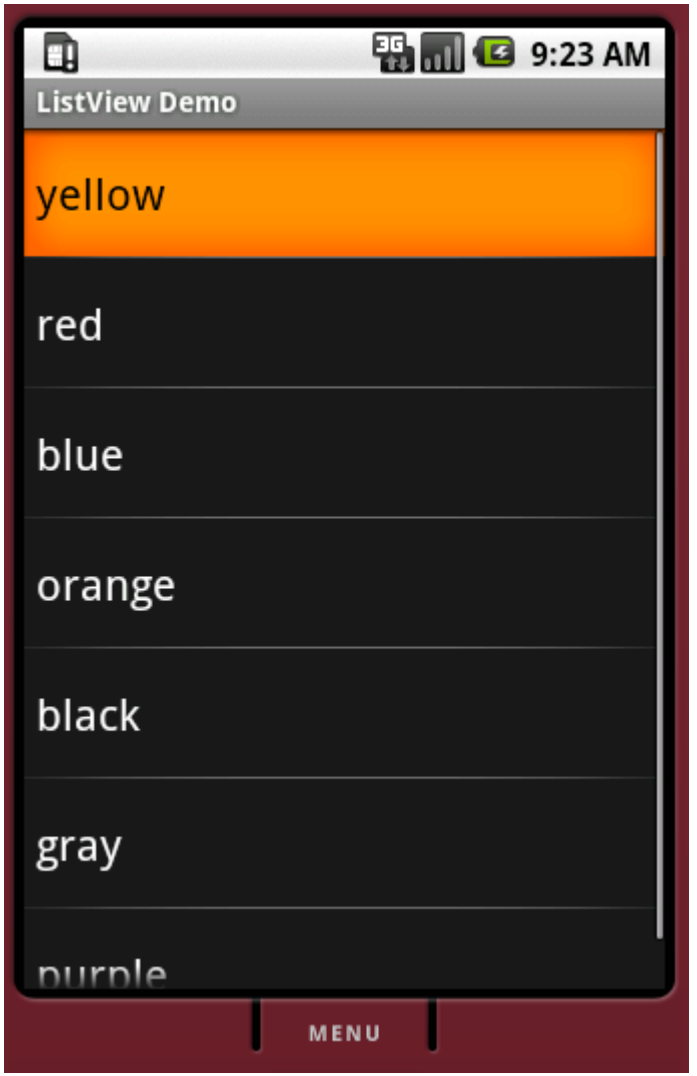
ListView

- The `ListView` control displays items in a vertical way.
- The simplest way for using the `ListView` class would be declaring a new class that extends `ListActivity` and set the list items by calling the `setListAdapter` method.

ListView

```
package com.abelski.android.samples;

import android.app.ListActivity;
import android.os.Bundle;
import android.widget.ArrayAdapter;

public class ListViewActivity extends ListActivity
{
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setListAdapter(new ArrayAdapter<String>(this,
            android.R.layout.simple_list_item_1, strings));
    }

    private String[] strings =
        { "yellow", "red", "blue", "orange", "black", "gray", "purple" };
}
```

ListView



GridView

- The `GridView` control displays items in a grid. These items can be strings, images, controls etc.

GridView

```
package com.abelski.android;

import android.app.Activity;
import android.content.Context;
import android.os.Bundle;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.GridView;
import android.widget.ImageView;


public class GridViewActivity extends Activity
{
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        GridView gridview = (GridView) findViewById(R.id.gridview);
        gridview.setAdapter(new ImagesAdapter(this));
    }
```

GridView

```
class ImagesAdapter extends BaseAdapter
{
    private Context context;

    public ImagesAdapter(Context c)
    {
        context = c;
    }

    public int getCount()
    {
        return images.length;
    }

    public Object getItem(int position)
    {
        return null;
    }

    public long getItemId(int position)
    {
        return (int)(position/3);
    }
```

GridView

```
public View getView(int position, View convertView, ViewGroup parent)
{
    ImageView imageView;
    if (convertView == null)
    {
        imageView = new ImageView(context);
    }
    else
    {
        imageView = (ImageView) convertView;
    }

    imageView.setImageResource(images[position]);
    return imageView;
}
```

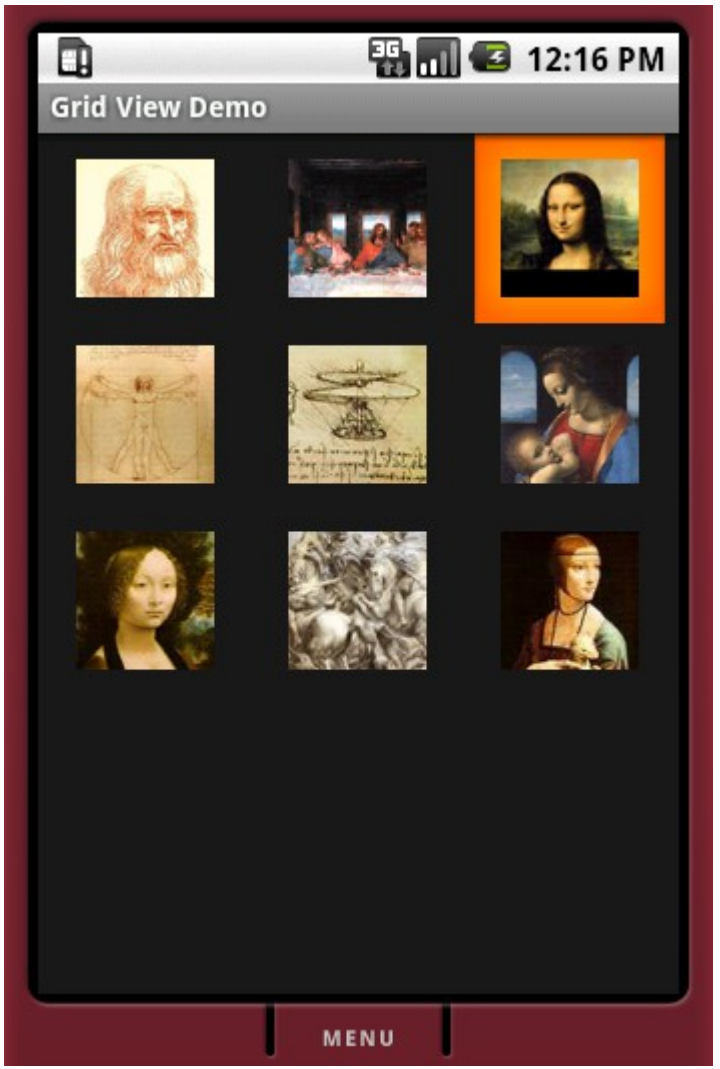
GridView

```
        private Integer[] images =
        {
            R.drawable.img_1, R.drawable.img_2, R.drawable.img_3,
            R.drawable.img_4, R.drawable.img_5, R.drawable.img_6,
            R.drawable.img_7, R.drawable.img_8, R.drawable.img_0
        };
    }
}
```

GridView

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

<GridView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/gridview" android:layout_width="fill_parent"
    android:layout_height="fill_parent" android:numColumns="auto_fit"
    android:verticalSpacing="8dp" android:horizontalSpacing="8dp"
    android:columnWidth="80dp" android:stretchMode="columnWidth"
    android:gravity="center" />
```

GridView

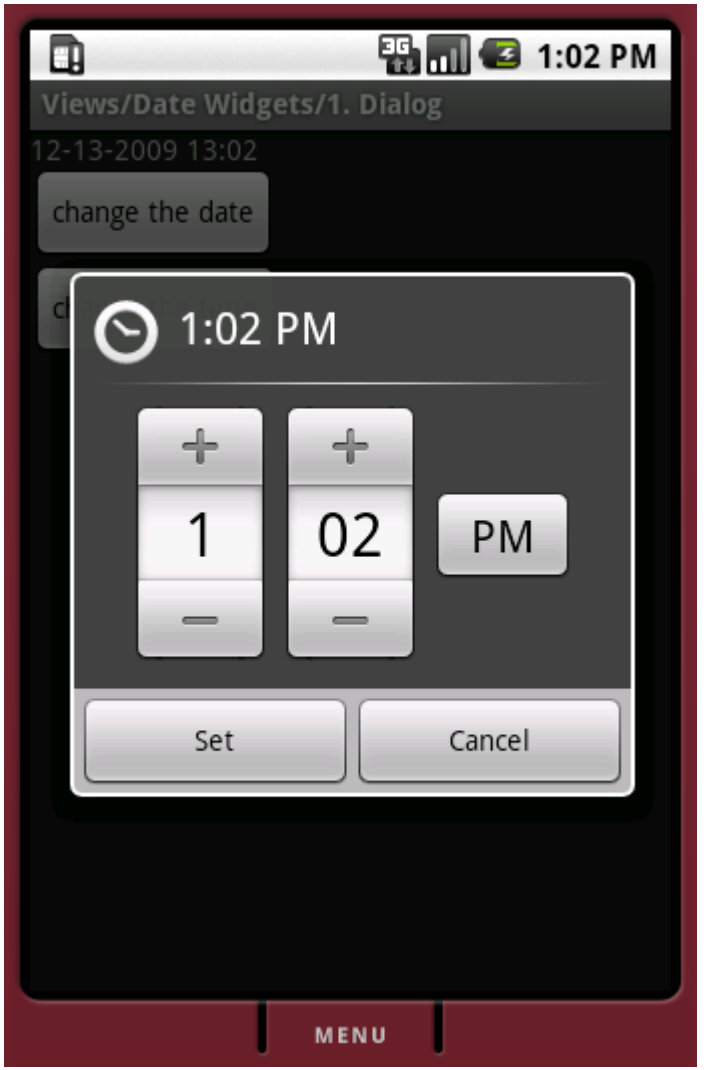


Date & Time Controls

- Android offers the following time-based controls:

  `DatePicker`, `TimePicker`, `AnalogClock` and `DigitalClock`.

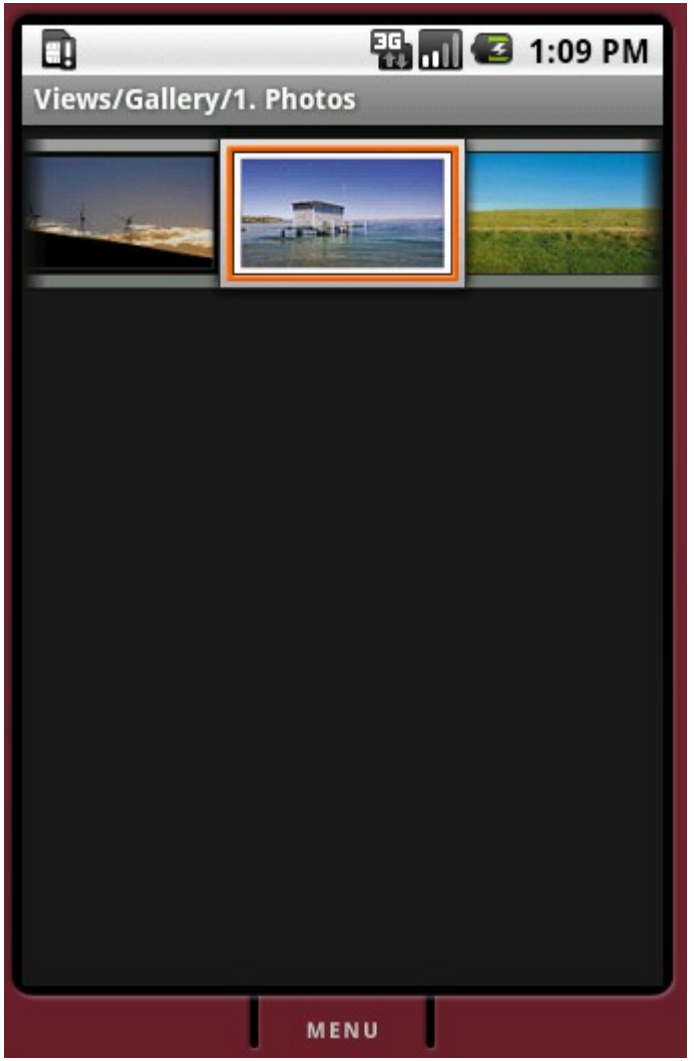
Date & Time Controls



Gallery

- Using the `Gallery` controller we can get horizontally scrollable list control that always focuses at the center of the list.
- The Gallery controller generally functions as a photo gallery.

Gallery



MapView

- Using the `MapView` controller we can get access to google maps.
- In order to use the `MapView` controller you first need to get an access key from google.

MapView

WebView

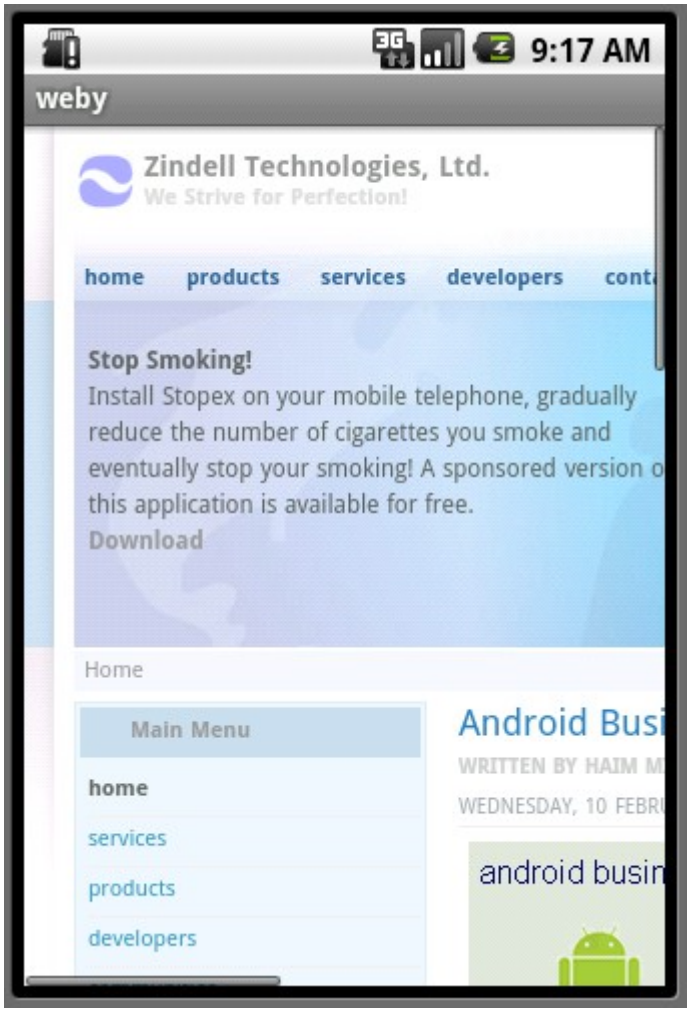
- The WebView controller functions as a mini browser that can show content on the web.
- The WebView controller uses the WebKit rendering engine.
- In order to use this controller we should add the internet user permission.

```
<uses-permission android:name="android.permission.INTERNET" />
```

WebView

```
package com.abelski.samples;

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

public class WebyActivity extends Activity
{

    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        WebView view = new WebView(this);
        view.loadUrl("http://www.zindell.com");
        setContentView(view);
    }
}
```

WebView



Typeface Fonts

- The android platform has several preconfigured typeface fonts such as '`sans`', '`monospace`' and '`serif`'. It is possible to add more fonts of our own.
- Various XML attributes allow us to customize the font our application uses.
- We set the typeface font we want to use using the `android:typeface` attribute.

Typeface Fonts

- When assigning `android:typeface` attribute with the '`normal`' value it defaults to the '`sans`' typeface.

Typeface Fonts

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent" android:layout_height="fill_parent"
    android:orientation="vertical" >

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" android:text="@string/sans"
        android:textSize="14sp" android:typeface="sans" />

    <TextView
        android:layout_width="fill_parent" android:layout_height="wrap_content"
        android:text="@string/serif" android:textSize="14sp"
        android:typeface="serif" />

    <TextView
        android:layout_width="fill_parent" android:layout_height="wrap_content"
        android:text="@string/monospace" android:textSize="14sp"
        android:typeface="monospace" />

</LinearLayout>
```

Typeface Fonts

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello World, SampleActivity!</string>
    <string name="sans">the sans font</string>
    <string name="monospace">the monospace font</string>
    <string name="serif">the serif font</string>
    <string name="app_name">fonts sample</string>
</resources>
```

Typeface Fonts

Text Style

- Using the `android:textStyle` attribute we can set a style. Possible values include `normal`, `bold` and `italic`. It is also possible setting `bold|italic`.

Text Style

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/normaltext" android:textSize="14sp"
        android:textStyle="normal"
        />

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/boldtext" android:textSize="14sp"
        android:textStyle="bold"
        />
```

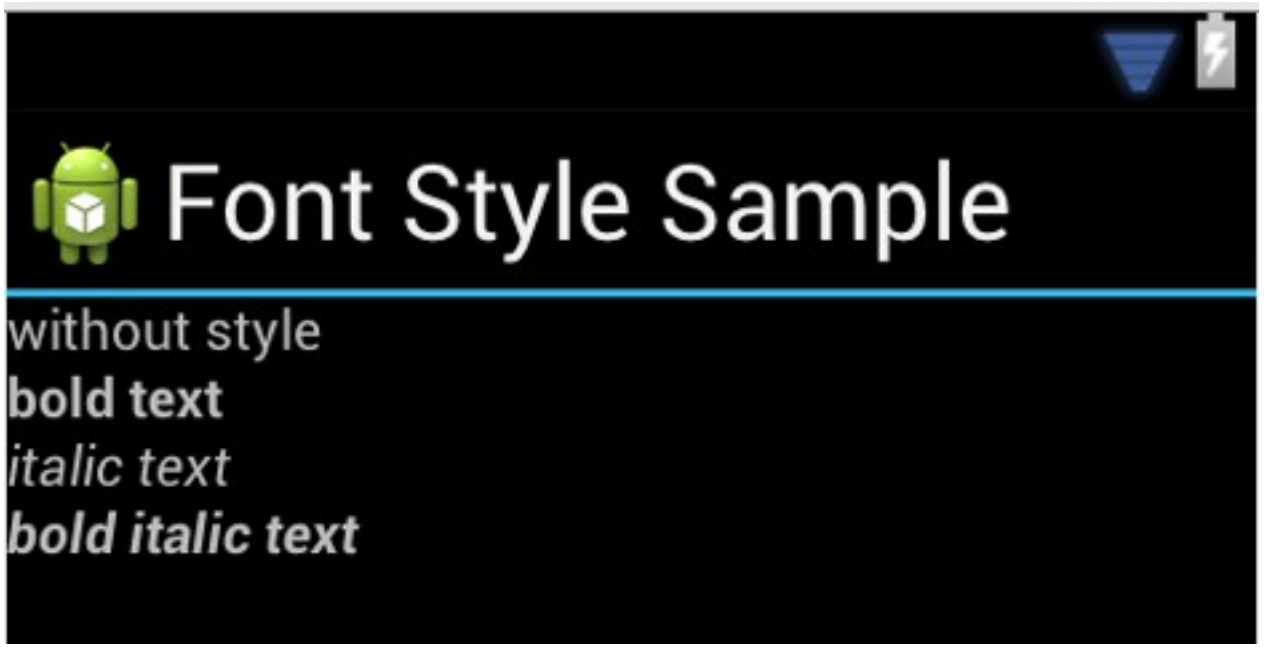
Text Style

```
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/italictext" android:textSize="14sp"
        android:textStyle="italic"
         />

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/bolditalictext" android:textSize="14sp"
        android:textStyle="bold|italic"
         />

</LinearLayout>
```

Text Style

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello World, SampleActivity!</string>
    <string name="normaltext">without style</string>
    <string name="boldtext">bold text</string>
    <string name="italictext">italic text</string>
    <string name="bolditalictext">bold italic text</string>
    <string name="app_name">Font Style Sample</string>
</resources>
```

Text Style



Text Size

- The `android:textSize` attribute specifies the font size. Its value includes two parts. The first is a floating-point number. The second is the unit.
- The possible units are: `sp` (scaled pixels), `px` (pixels), `dp` (density independent pixels), `in` (inches) and `mm` (millimeters).

Text Size

- The scaled pixels (`sp`) unit is the same as density independent pixels (`dp`) with one difference. When using scaled pixels (`sp`) the font size is also influenced by the user android platform settings.
- Google recommends that we use the scaled pixels dimension.

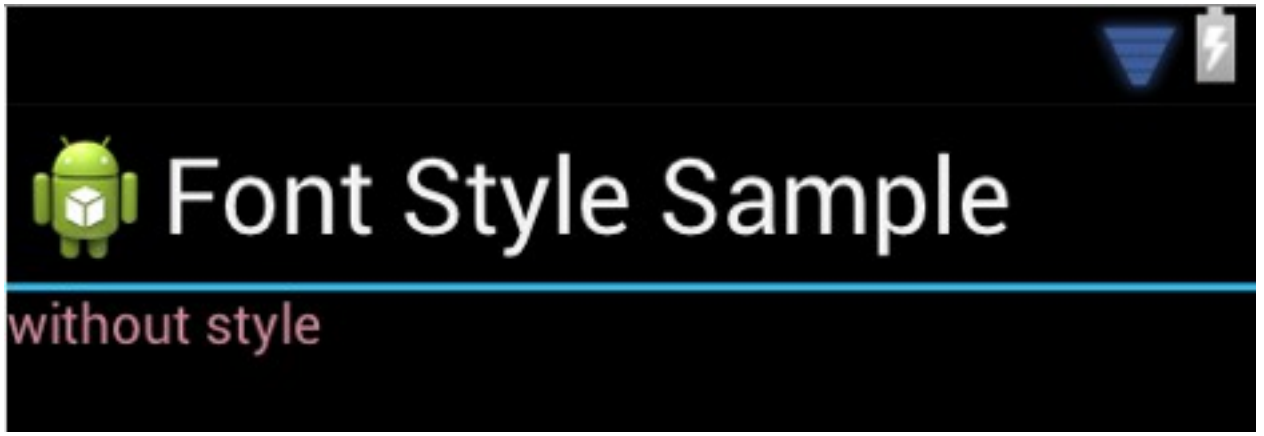
Text Color

- We can set the text color using the `android:textColor` attribute.
- The value is a hexadecimal RGB value with an optional alpha channel. Similarly to CSS.

Text Color

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <TextView
          android:layout_width="fill_parent"
          android:layout_height="wrap_content"
          android:text="@string/normaltext"
          android:textSize="14sp"
          android:textColor="#cc8899"
          android:textStyle="normal" />

</LinearLayout>
```

Text Color



Text Shadow

- We can easily add some shadow to our texts by using the `android:shadowColor`, `android:shadowRadius`, `android:shadowDx` and `android:shadowDy`.
- The `android:shadowRadius` attribute specifies the radius of the shadow. The value should be a floating point number. The bigger the radius is so the shadow will be blurred.

Text Shadow

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/normaltext"
        android:shadowColor="#22bbee"
        android:shadowRadius="3.5"
        android:shadowDx="7"
        android:shadowDy="3"
        android:textSize="44dp"/>

</LinearLayout>
```

Text Shadow

The visual designer doesn't support text shadow.

Using Custom Fonts

- We can easily use a typeface font the android platform isn't preconfigured with by adding the new font to the assets folder and calling the `Typeface.createFromAsset` static function.

Using Custom Fonts

```
public class SampleActivity extends Activity
{
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        TextView text = (TextView)findViewById(R.id.text);
        Typeface font = Typeface.createFromAsset(
          getAssets(), "BaroqueScript.ttf");
        text.setTypeface(font);
    }
}
```

Using Custom Fonts

The visual designer doesn't support custom fonts.

User Interface

Introduction

- Android provides a simple framework with “out of the box” controls we can use.
- All available controls are based on two parent classes: `View` and `ViewGroup`. `ViewGroup` extends `View`.

Introduction

- The android platform enables developing the GUI in several different approaches.
- One approach is based on coding everything in our source code.
- Another approach is based on defining the user interface in XML document.
- A third approach includes both defining the user interface in XML document and referring it from within the source code.

View, Widget, Control Each of these represents a user interface element, such as a button, a grid, a list, a window, a dialog box etc. The terms “view,” “widget,” and “control” are used interchangeably. They mean the same thing.

Container is a view used to contain other views. One example is a grid that is used as a container as it contains cells. Each cell is a view.

The layout is an XML document that describes a view.

GUI (in Source Code) Sample

```
package com.abelski.android;

import android.app.Activity;
import android.os.Bundle;
import android.view.ViewGroup.LayoutParams;
import android.widget.LinearLayout;
import android.widget.TextView;

public class SimpleGUIInCode extends Activity
{
    private LinearLayout firstNameContainer;
    private LinearLayout lastNameContainer;
    private LinearLayout emailContainer;
    private LinearLayout addressContainer;
    private LinearLayout parentContainer;

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

GUI (in Source Code) Sample

```
    createEmailContainer();
    createAddressContainer();
    createParentContainer();
    setContentView(parentContainer);
}

private void createFirstNameContainer()
{
    firstNameContainer = new LinearLayout(this);
    firstNameContainer.setLayoutParams(new LayoutParams(
            LayoutParams.FILL_PARENT, LayoutParams.WRAP_CONTENT));
    firstNameContainer.setOrientation(LinearLayout.HORIZONTAL);
    TextView firstNameLabel = new TextView(this);
    firstNameLabel.setText("First Name: ");
    firstNameContainer.addView(firstNameLabel);
    TextView firstNameValue = new TextView(this);
    firstNameValue.setText("Haim");
    firstNameContainer.addView(firstNameValue);
}
```

GUI (in Source Code) Sample

```
private void createLastNameContainer()
{
    lastNameContainer = new LinearLayout(this);
    lastNameContainer.setLayoutParams(new LayoutParams(
            LayoutParams.FILL_PARENT, LayoutParams.WRAP_CONTENT));
    lastNameContainer.setOrientation(LinearLayout.HORIZONTAL);
    TextView lastNameLabel = new TextView(this);
    lastNameLabel.setText("Last Name: ");
    lastNameContainer.addView(lastNameLabel);
    TextView lastNameValue = new TextView(this);
    lastNameValue.setText("Michael");
    lastNameContainer.addView(lastNameValue);
}

private void createEmailContainer()
{
    emailContainer = new LinearLayout(this);
    emailContainer.setLayoutParams(new LayoutParams(
            LayoutParams.FILL_PARENT, LayoutParams.WRAP_CONTENT));
    emailContainer.setOrientation(LinearLayout.HORIZONTAL);
    TextView emailLabel = new TextView(this);
    emailLabel.setText("Email Address:");
```

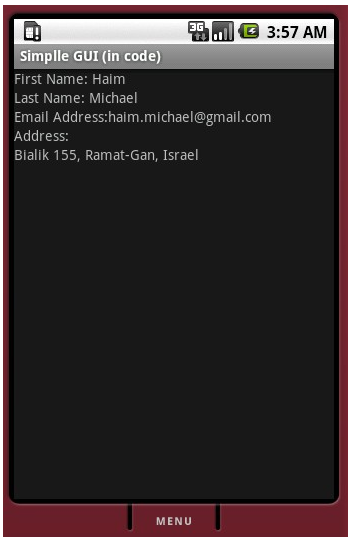
GUI (in Source Code) Sample

```
    TextView emailValue = new TextView(this);
    emailValue.setText("haim.michael@gmail.com");
    emailContainer.addView(emailLabel);
    emailContainer.addView(emailValue);
}

private void createAddressContainer()
{
    addressContainer = new LinearLayout(this);
    addressContainer.setLayoutParams(new LayoutParams(
            LayoutParams.FILL_PARENT, LayoutParams.WRAP_CONTENT));
    addressContainer.setOrientation(LinearLayout.VERTICAL);
    TextView addressLabel = new TextView(this);
    addressLabel.setText("Address:");
    TextView addressValue = new TextView(this);
    addressValue.setText("Bialik 155, Ramat-Gan, Israel");
    addressContainer.addView(addressLabel);
    addressContainer.addView(addressValue);
}
```

GUI (in Source Code) Sample

```
    private void createParentContainer()
    {
        parentContainer = new LinearLayout(this);
        parentContainer.setLayoutParams(new LayoutParams(
                LayoutParams.FILL_PARENT, LayoutParams.FILL_PARENT));
        parentContainer.setOrientation(LinearLayout.VERTICAL);
        parentContainer.addView(firstNameContainer);
        parentContainer.addView(lastNameContainer);
        parentContainer.addView(emailContainer);
        parentContainer.addView(addressContainer);
    }
}
```

GUI (in Source Code) Sample



GUI (in XML) Sample

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

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

    <!-- FIRST NAME CONTAINER -->
    <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation="horizontal" android:layout_width="fill_parent"
        android:layout_height="wrap_content">
        <TextView android:layout_width="wrap_content"
            android:layout_height="wrap_content" android:text="First Name:" />
        <TextView android:layout_width="wrap_content"
            android:layout_height="wrap_content" android:text="Haim" />
    </LinearLayout>
```

GUI (in XML) Sample

```
<!-- LAST NAME CONTAINER -->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal" android:layout_width="fill_parent"
    android:layout_height="wrap_content">
    <TextView android:layout_width="wrap_content"
        android:layout_height="wrap_content" android:text="Last Name:" />
    <TextView android:layout_width="wrap_content"
        android:layout_height="wrap_content" android:text="Michael" />
</LinearLayout>

<!-- EMAIL ADDRESS CONTAINER -->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal" android:layout_width="fill_parent"
    android:layout_height="wrap_content">
    <TextView android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Email Address:" />
    <TextView android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Haim.Michael@gmail.com " />
</LinearLayout>
```

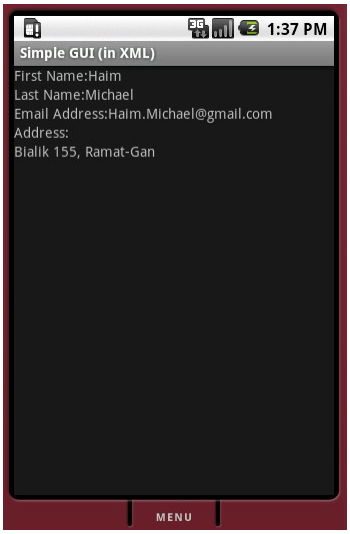
GUI (in XML) Sample

```
    <!-- ADDRESS CONTAINER -->
    <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation="vertical" android:layout_width="fill_parent"
        android:layout_height="wrap_content">
        <TextView android:layout_width="fill_parent"
            android:layout_height="wrap_content" android:text="Address: " />
        <TextView android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Bialik 155, Ramat-Gan" />
    </LinearLayout>

</LinearLayout>
```

GUI (in XML) Sample

```
package com.abelski.android;

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

public class SimpleGUIInXML extends Activity {

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

GUI (in XML) Sample



GUI (in XML & Code) Sample

```
<?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">

    <!-- FIRST NAME CONTAINER -->
    <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation="horizontal" android:layout_width="fill_parent"
        android:layout_height="wrap_content">
        <TextView android:id="@+id/firstNameTextLabel"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"/>
        <TextView android:id="@+id/firstNameTextValue"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />
    </LinearLayout>

<!-- LAST NAME CONTAINER -->
    <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation="horizontal" android:layout_width="fill_parent"
        android:layout_height="wrap_content">
```

GUI (in XML & Code) Sample

```
    <TextView android:id="@+id/lastNameTextLabel"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>
    <TextView android:id="@+id/lastNameTextValue"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
</LinearLayout>

<!-- EMAIL CONTAINER -->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal" android:layout_width="fill_parent"
    android:layout_height="wrap_content">
    <TextView android:id="@+id/emailTextLabel"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>
    <TextView android:id="@+id/emailTextValue"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
</LinearLayout>
```

GUI (in XML & Code) Sample

```
    <!-- ADDRESS CONTAINER -->
    <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation="vertical" android:layout_width="fill_parent"
        android:layout_height="wrap_content">
        <TextView android:id="@+id/addressLabelText"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content" />
        <TextView android:id="@+id/addressValueText"
            android:layout_height="wrap_content"
            android:layout_width="fill_parent" />
    </LinearLayout>

</LinearLayout>
```

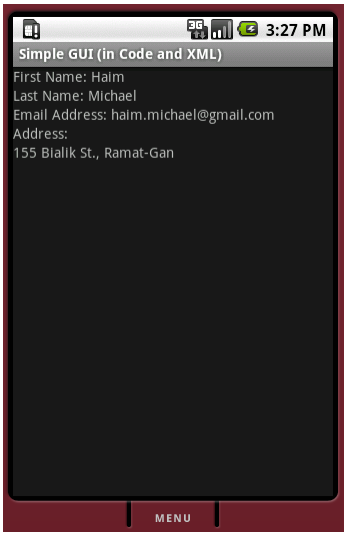
GUI (in XML & Code) Sample

```
package com.abelski.android;

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

public class SimpleGUIinCodeXML extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        TextView firstNameLabel = (TextView)findViewById(R.id.firstNameTextLabel);
        firstNameLabel.setText("First Name: ");
        TextView firstNameValue = (TextView)findViewById(R.id.firstNameTextValue);
        firstNameValue.setText("Haim");
        TextView lastNameLabel = (TextView)findViewById(R.id.lastNameTextLabel);
```

GUI (in XML & Code) Sample

```
        lastNameLabel.setText("Last Name: ");
        TextView lastNameValue = (TextView)findViewById(R.id.lastNameTextValue);
        lastNameValue.setText("Michael");
        TextView emailLabel = (TextView)findViewById(R.id.emailTextLabel);
        emailLabel.setText("Email Address: ");
        TextView emailValue = (TextView)findViewById(R.id.emailTextValue);
        emailValue.setText("haim.michael@gmail.com");
        TextView addressLabel = (TextView)findViewById(R.id.addressLabelText);
        addressLabel.setText("Address: ");
        TextView addressValue = (TextView)findViewById(R.id.addressValueText);
        addressValue.setText("155 Bialik St., Ramat-Gan");
    }
}
```

GUI (in XML & Code) Sample



TextView

- Present text without allowing the user to edit it.
- Using the `Linkify` class we can cause the android platform to automatically present any URL address, any email address and any telephone number as one the user can easily clicks on and navigates to.

```
TextView addressValue =
      (TextView)findViewById(R.id.addressValueText);
addressValue.setText("The website is www.zindell.com, the email
      address is info@zindell.com and the telephone number
      is +972-54-6655837.");
Linkify.addLinks(addressValue, Linkify.ALL);
```


Clicking a link will automatically cause the default intent to be called accordingly. This way (for example) clicking a phone number will automatically open the phone dialer with that number and calling a URL address will automatically open the browser.

TextView

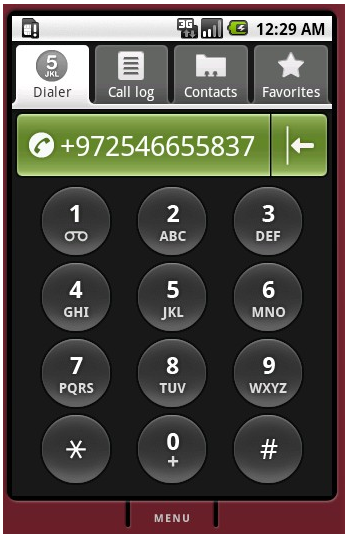
```
package com.abelski.android;

import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;
import android.text.util.Linkify;

public class SimpleTextViewSample extends Activity
{
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        TextView addressValue =
            (TextView)findViewById(R.id.addressValueText);
        addressValue.setText("The website is www.zindell.com, the email
            address is info@zindell.com and the telephone number is +972-
            54-6655837.");
        Linkify.addLinks(addressValue, Linkify.ALL);
    }
}
```

TextView

```
<?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">

    <LinearLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation="vertical" android:layout_width="fill_parent"
        android:layout_height="wrap_content">
        <TextView android:id="@+id/addressValueText"
            android:layout_height="wrap_content"
            android:layout_width="fill_parent" />
    </LinearLayout>

</LinearLayout>
```

TextView

TextView



TextView

- When using the strings as resources (defined within the `strings.xml` file) we can add markups to specify the required style.

View, Widget, Control Each of these represents a user interface element, such as a button, a grid, a list, a window, a dialog box etc. The terms “view,” “widget,” and “control” are used interchangeably. They mean the same thing.

Container is a view used to contain other views. One example is a grid that is used as a container as it contains cells. Each cell is a view.

The layout is an XML document that describes a view.

TextView

```
package com.abelski.android;

import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;
import android.text.util.Linkify;

public class SimpleTextViewSample extends Activity
{
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        TextView addressValue =
            (TextView)findViewById(R.id.addressValueText);
        addressValue.setText(R.string.address_str);
        Linkify.addLinks(addressValue, Linkify.ALL);
    }
}
```

TextView

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello World, SimpleGUIinCodeXML!</string>
    <string name="app_name">Simple GUI (in Code and XML)</string>
    <string name="address_str">The <b>website</b> is www.zindell.com, the
        <i>email</i> address is info@zindell.com and the telephone number is
        +972-54-6655837.</string>
</resources>
```

TextView

```
<?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">

    <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation="vertical" android:layout_width="fill_parent"
        android:layout_height="wrap_content">
        <TextView android:id="@+id/addressValueText"
            android:layout_height="wrap_content"
            android:layout_width="fill_parent" />
    </LinearLayout>

</LinearLayout>
```

TextView Style Sample

EditText

- The `EditText` class extends `TextView`. Unlike `TextView`, the `EditText` control allows the user to edit the text.
- The `EditText` control has various properties that configures its functionality.

  Using the `autoText` property we can cause it to automatically fixes spelling mistakes. Using the `capitalize` property we can makes it capitalizing first sentence letters. Using the `phoneNumber` property we can get a control that accepts phone numbers only. Using the `singleLine` property the user will be forced to enter a one line input.

EditText

- We can change the style of specific portions of the text by applying static markups directly to the strings (within the `strings.xml` resource file) the same way we can do it with TextView objects.
- We can alternatively apply the required style by using the `Spannable` class.

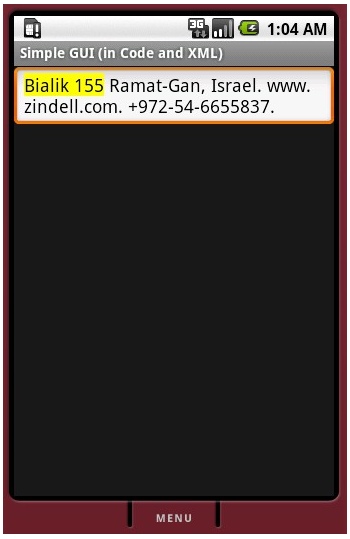
EditText

```
package com.abelski.android;

import android.app.Activity;
import android.graphics.Color;
import android.os.Bundle;
import android.widget.EditText;
import android.text.Spannable;
import android.text.style.BackgroundColorSpan;

public class SimpleGUISample extends Activity
{
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        EditText addressValue = (EditText)
                findViewById(R.id.addressValueText);
        addressValue.setText(R.string.address_str);
        Spannable span = addressValue.getEditableText();
        span.setSpan(new BackgroundColorSpan(Color.YELLOW), 0, 10, 0);
    }
}
```

EditText

```
<?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">

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

        <EditText android:id="@+id/addressValueText"
            android:layout_height="wrap_content"
            android:layout_width="fill_parent" />

    </LinearLayout>

</LinearLayout>
```

EditText



AutoCompleteTextView

- The `AutoCompleteTextView` class is a `TextView` with the autocomplete functionality.
- The possible texts are set using an `ArrayAdapter` object.

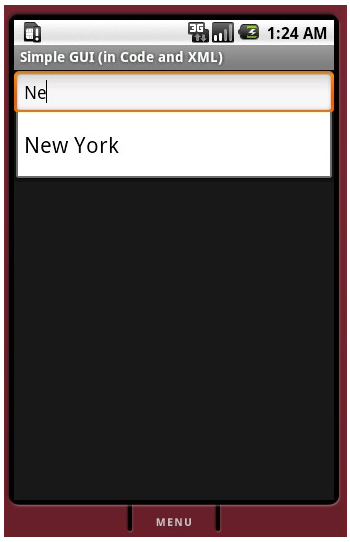
AutoCompleteTextView

```
package com.abelski.android;

import android.app.Activity;
import android.os.Bundle;
import android.widget.ArrayAdapter;
import android.widget.AutoCompleteTextView;

public class SimpleGUISample extends Activity
{

    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        AutoCompleteTextView city =
            (AutoCompleteTextView) findViewById(R.id.city);
        ArrayAdapter<String> adapter =
            new ArrayAdapter<String>(this,
            android.R.layout.simple_dropdown_item_1line,
            new String[] {"New York", "Tel Aviv", "Moscow", "Jerusalem",
            "Beirut", "London" });
        city.setAdapter(adapter);
    }
}
```

AutoCompleteTextView

```
<?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">

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

    <AutoCompleteTextView android:id="@+id/city"
        android:layout_height="wrap_content"
        android:layout_width="fill_parent" />

</LinearLayout>

</LinearLayout>
```

AutoCompleteTextView



MultiAutoCompleteTextView

- The `MultiAutoCompleteTextView` is an improvement to the `AutoCompleteTextView`.
- It allows us having the auto complete feature for more than one part of the text view.

MultiAutoCompleteTextView

```
<?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">

    <LinearLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation="vertical" android:layout_width="fill_parent"
        android:layout_height="wrap_content">
        <MultiAutoCompleteTextView android:id="@+id/colors"
            android:layout_height="wrap_content"
            android:layout_width="fill_parent" />
    </LinearLayout>

</LinearLayout>
```

MultiAutoCompleteTextView

Button

- The `Button` control is the simplest available button on the Android platform.
- Registering an events handler is done by calling the `setOnClickListener` function on the `Button` object and passing an `OnClickListener` object to it.

Button

```
package com.abelski.android;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.MultiAutoCompleteTextView;
import android.widget.TextView;


public class SimpleGUISample extends Activity
{
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button btPlus = (Button)this.findViewById(R.id.btPlus);
```

Button

```
        btPlus.setOnClickListener(new OnClickListener()
        {
            public void onClick(View v)
            {
                double firstNum =
                    Double.parseDouble(
                        ((EditText)SimpleGUISample.this.
                        findViewById(R.id.num_1))
                        .getText().toString());
                double secondNum =  Double.parseDouble(
                        ((EditText)SimpleGUISample.this.
                        findViewById(R.id.num_2)).getText().toString());
                double sum = firstNum + secondNum;
                TextView total = ((TextView)SimpleGUISample.
                        this.findViewById(R.id.sum));
                total.setText(String.valueOf(sum));
            }
        });
    }
}
```

Button

```
<?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="wrap_content">

    <EditText android:id="@+id/num_1" android:inputType="numberDecimal"
        android:layout_height="wrap_content"
        android:layout_width="fill_parent" />
    <EditText android:id="@+id/num_2" android:inputType="numberDecimal"
        android:layout_height="wrap_content"
        android:layout_width="fill_parent" />
    <Button android:id="@+id/btPlus" android:text="+"
        android:layout_height="wrap_content"
        android:layout_width="fill_parent" />
    <EditText android:id="@+id/sum" android:inputType="numberDecimal"
        android:editable="false"
        android:layout_height="wrap_content"
        android:layout_width="fill_parent" />

</LinearLayout>
```

Button

ImageButton

- The `ImageButton` control is similar to `Button`. The image can be set either dynamically by calling `setImageResource` method on the `ImageButton` object or by specifying the image in the XML layout file.
- Handling the events is as with a simple `Button`.

ImageButton

```
package com.abelski.android;

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


public class SimpleGUISample extends Activity
{
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        ImageButton btPlus = (ImageButton)this.findViewById(R.id.btPlus);
        //btPlus.setImageResource(R.drawable.plus);
        btPlus.setOnClickListener(new OnClickListener()
        {
```

ImageButton

```
            public void onClick(View v)
            {
                double firstNum =
                    Double.parseDouble(((EditText)SimpleGUISample.
                    this.findViewById(R.id.num_1)).getText().toString());
                double secondNum =
                    Double.parseDouble(((EditText)SimpleGUISample.
                    this.findViewById(R.id.num_2)).getText().toString());

                double sum = firstNum + secondNum;

                TextView total = ((TextView)SimpleGUISample.
                    this.findViewById(R.id.sum));

                total.setText(String.valueOf(sum));
            }
        });
    }
}
```

ImageButton

```
<?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="wrap_content">

    <EditText android:id="@+id/num_1" android:inputType="numberDecimal"
        android:layout_height="wrap_content"
        android:layout_width="fill_parent" />
    <EditText android:id="@+id/num_2" android:inputType="numberDecimal"
        android:layout_height="wrap_content"
        android:layout_width="fill_parent" />
    <ImageButton android:id="@+id/btPlus" android:src="@drawable/plus"
        android:layout_height="wrap_content"
        android:layout_width="fill_parent" />
    <EditText android:id="@+id/sum" android:inputType="numberDecimal"
        android:editable="false"
        android:layout_height="wrap_content"
        android:layout_width="fill_parent" />

</LinearLayout>
```

ImageButton

ToggleButton

- The `ToggleButton` control is kind of a check box. It is a two state button ('on' and 'off').
- Default behavior sets the text on this type of button either 'On' or 'Off'.
- It is possible to change the texts (instead of 'On' and 'Off') using the 'android:textOn' and 'android:textOff' properties.

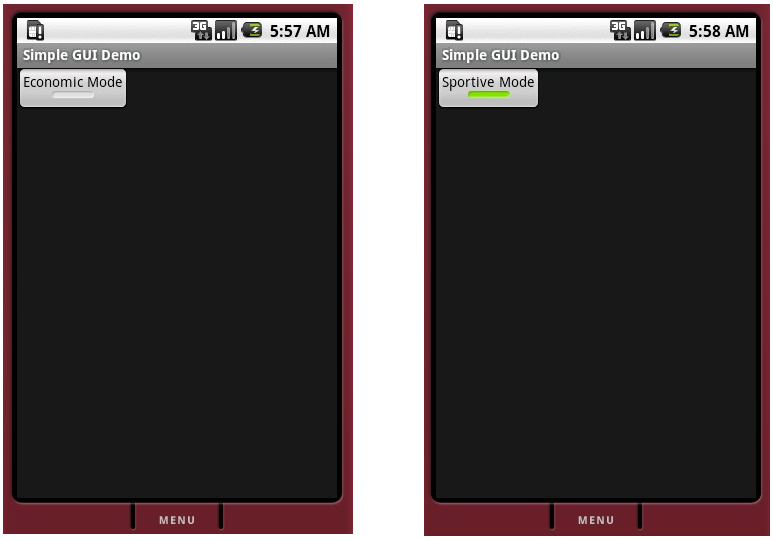
ToggleButton

```
<?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="wrap_content">

<ToggleButton android:id="@+id/bt"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:textOn="Sportive Mode"
android:textOff="Economic Mode"
android:text="Automatic Gear"/>

</LinearLayout>
```

ToggleButton

```
package com.abelski.android;

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

public class SimpleGUISample extends Activity
{
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

ToggleButton



CheckBox

- The `CheckBox` control is a simple check box similar to those we can find in other software development platforms.
- It is a two state button that allows the user to toggle its state.

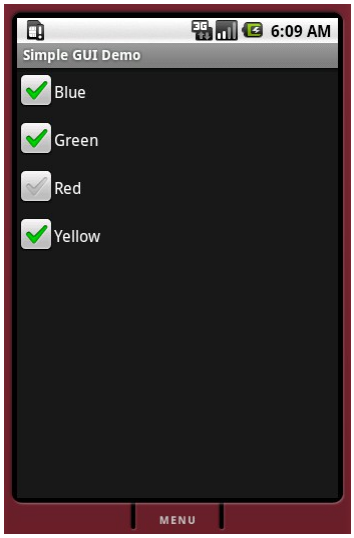
CheckBox

```
<?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="wrap_content">

<CheckBox android:text="Blue" android:layout_width="wrap_content"
android:layout_height="wrap_content" />
<CheckBox android:text="Green" android:layout_width="wrap_content"
android:layout_height="wrap_content" />
<CheckBox android:text="Red" android:layout_width="wrap_content"
android:layout_height="wrap_content" />
<CheckBox android:text="Yellow" android:layout_width="wrap_content"
android:layout_height="wrap_content" />

</LinearLayout>
```

CheckBox

```
package com.abelski.android;

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

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

CheckBox



RadioButton

- The `RadioButton` control allows selecting one option only.
- In order to allow selecting only one of the options we should place them within the same group.

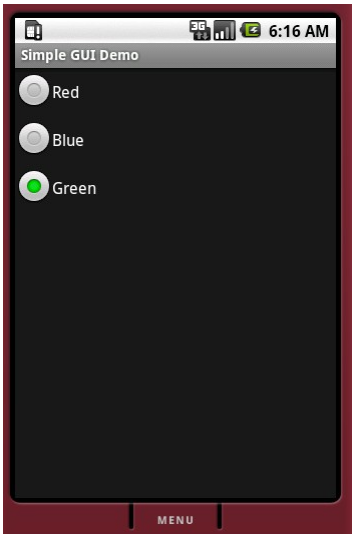
RadioButton

```
<?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="wrap_content">

<RadioGroup android:layout_width="wrap_content"
android:layout_height="wrap_content">

    <RadioButton android:id="@+id/redRBtn" android:text="Red"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <RadioButton android:id="@+id/blueRBtn" android:text="Blue"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <RadioButton android:id="@+id/greenRBtn" android:text="Green"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

</RadioGroup>

</LinearLayout>
```

RadioButton

```
package com.abelski.android;

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

public class SimpleGUISample extends Activity
{
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

RadioButton



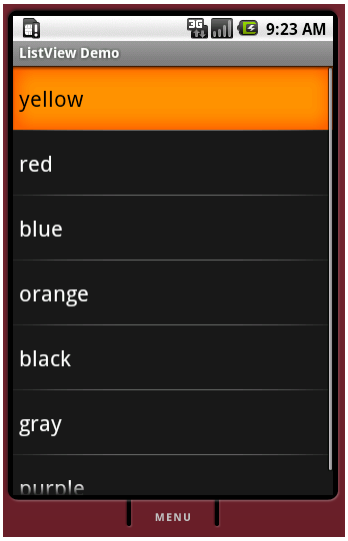
ListView

- The `ListView` control displays items in a vertical way.
- The simplest way for using the `ListView` class would be declaring a new class that extends `ListActivity` and set the list items by calling the `setListAdapter` method.

ListView

```
package com.abelski.android.samples;

import android.app.ListActivity;
import android.os.Bundle;
import android.widget.ArrayAdapter;

public class ListViewActivity extends ListActivity
{
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setListAdapter(new ArrayAdapter<String>(this,
            android.R.layout.simple_list_item_1, strings));
    }

    private String[] strings =
        { "yellow", "red", "blue", "orange", "black", "gray", "purple" };
}
```

ListView



GridView

- The `GridView` control displays items in a grid. These items can be strings, images, controls etc.

GridView

```
package com.abelski.android;

import android.app.Activity;
import android.content.Context;
import android.os.Bundle;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.GridView;
import android.widget.ImageView;


public class GridViewActivity extends Activity
{
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        GridView gridview = (GridView) findViewById(R.id.gridview);
        gridview.setAdapter(new ImagesAdapter(this));
    }
```

GridView

```
class ImagesAdapter extends BaseAdapter
{
    private Context context;

    public ImagesAdapter(Context c)
    {
        context = c;
    }

    public int getCount()
    {
        return images.length;
    }

    public Object getItem(int position)
    {
        return null;
    }

    public long getItemId(int position)
    {
        return (int)(position/3);
    }
```

GridView

```
public View getView(int position, View convertView, ViewGroup parent)
{
    ImageView imageView;
    if (convertView == null)
    {
        imageView = new ImageView(context);
    }
    else
    {
        imageView = (ImageView) convertView;
    }

    imageView.setImageResource(images[position]);
    return imageView;
}
```

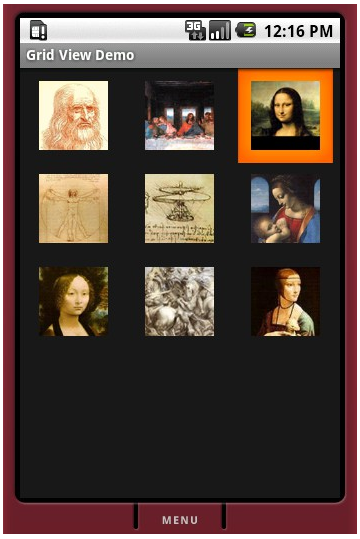
GridView

```
        private Integer[] images =
        {
            R.drawable.img_1, R.drawable.img_2, R.drawable.img_3,
            R.drawable.img_4, R.drawable.img_5, R.drawable.img_6,
            R.drawable.img_7, R.drawable.img_8, R.drawable.img_0
        };
    }
}
```

GridView

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

<GridView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/gridview" android:layout_width="fill_parent"
    android:layout_height="fill_parent" android:numColumns="auto_fit"
    android:verticalSpacing="8dp" android:horizontalSpacing="8dp"
    android:columnWidth="80dp" android:stretchMode="columnWidth"
    android:gravity="center" />
```

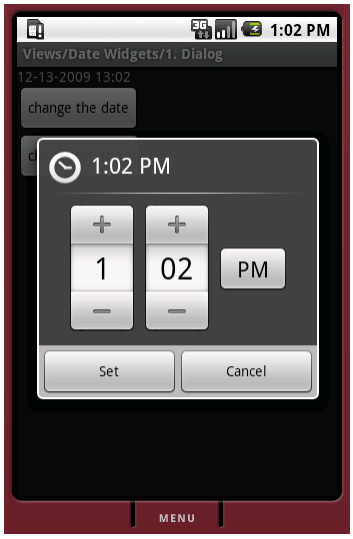
GridView



Date & Time Controls

- Android offers the following time-based controls: `DatePicker`, `TimePicker`, `AnalogClock` and `DigitalClock`.

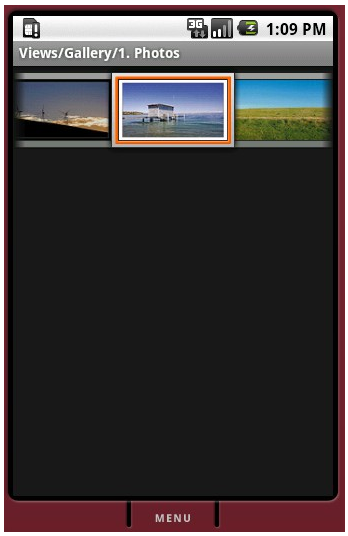
Date & Time Controls



Gallery

- Using the `Gallery` controller we can get horizontally scrollable list control that always focuses at the center of the list.
- The Gallery controller generally functions as a photo gallery.

Gallery



MapView

- Using the `MapView` controller we can get access to google maps.
- In order to use the `MapView` controller you first need to get an access key from google.

MapView

WebView

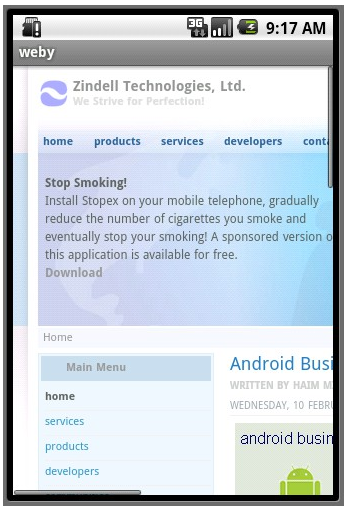
- The WebView controller functions as a mini browser that can show content on the web.
- The WebView controller uses the WebKit rendering engine.
- In order to use this controller we should add the internet user permission.

```
<uses-permission android:name="android.permission.INTERNET" />
```

WebView

```
package com.abelski.samples;

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

public class WebyActivity extends Activity
{

    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        WebView view = new WebView(this);
        view.loadUrl("http://www.zindell.com");
        setContentView(view);
    }
}
```

WebView



Typeface Fonts

- The android platform has several preconfigured typeface fonts such as '`sans`', '`monospace`' and '`serif`'. It is possible to add more fonts of our own.
- Various XML attributes allow us to customize the font our application uses.
- We set the typeface font we want to use using the `android:typeface` attribute.

Typeface Fonts

- When assigning `android:typeface` attribute with the '`normal`' value it defaults to the '`sans`' typeface.

Typeface Fonts

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent" android:layout_height="fill_parent"
    android:orientation="vertical" >

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" android:text="@string/sans"
        android:textSize="14sp" android:typeface="sans" />

    <TextView
        android:layout_width="fill_parent" android:layout_height="wrap_content"
        android:text="@string/serif" android:textSize="14sp"
        android:typeface="serif" />

    <TextView
        android:layout_width="fill_parent" android:layout_height="wrap_content"
        android:text="@string/monospace" android:textSize="14sp"
        android:typeface="monospace" />

</LinearLayout>
```

Typeface Fonts

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello World, SampleActivity!</string>
    <string name="sans">the sans font</string>
    <string name="monospace">the monospace font</string>
    <string name="serif">the serif font</string>
    <string name="app_name">fonts sample</string>
</resources>
```

Typeface Fonts

Text Style

- Using the `android:textStyle` attribute we can set a style. Possible values include `normal`, `bold` and `italic`. It is also possible setting `bold|italic`.

Text Style

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/normaltext" android:textSize="14sp"
        android:textStyle="normal"
        />

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/boldtext" android:textSize="14sp"
        android:textStyle="bold"
        />
```

Text Style

```
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/italictext" android:textSize="14sp"
        android:textStyle="italic"
        />

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/bolditalictext" android:textSize="14sp"
        android:textStyle="bold|italic"
        />

</LinearLayout>
```

Text Style

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello World, SampleActivity!</string>
    <string name="normaltext">without style</string>
    <string name="boldtext">bold text</string>
    <string name="italictext">italic text</string>
    <string name="bolditalictext">bold italic text</string>
    <string name="app_name">Font Style Sample</string>
</resources>
```

Text Style

Text Size

- The `android:textSize` attribute specifies the font size. Its value includes two parts. The first is a floating-point number. The second is the unit.
- The possible units are: `sp` (scaled pixels), `px` (pixels), `dp` (density independent pixels), `in` (inches) and `mm` (millimeters).

Text Size

- The scaled pixels (`sp`) unit is the same as density independent pixels (`dp`) with one difference. When using scaled pixels (`sp`) the font size is also influenced by the user android platform settings.
- Google recommends that we use the scaled pixels dimension.

Text Color

- We can set the text color using the `android:textColor` attribute.
- The value is a hexadecimal RGB value with an optional alpha channel. Similarly to CSS.

Text Color

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <TextView
          android:layout_width="fill_parent"
          android:layout_height="wrap_content"
          android:text="@string/normaltext"
          android:textSize="14sp"
          android:textColor="#cc8899"
          android:textStyle="normal" />

</LinearLayout>
```

Text Color

Text Shadow

- We can easily add some shadow to our texts by using the `android:shadowColor`, `android:shadowRadius`, `android:shadowDx` and `android:shadowDy`.
- The `android:shadowRadius` attribute specifies the radius of the shadow. The value should be a floating point number. The bigger the radius is so the shadow will be blurred.

Text Shadow

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/normaltext"
        android:shadowColor="#22bbee"
        android:shadowRadius="3.5"
        android:shadowDx="7"
        android:shadowDy="3"
        android:textSize="44dp"/>

</LinearLayout>
```

Text Shadow

The visual designer doesn't support text shadow.

Using Custom Fonts

- We can easily use a typeface font the android platform isn't preconfigured with by adding the new font to the assets folder and calling the `Typeface.createFromAsset` static function.

Using Custom Fonts

```
public class SampleActivity extends Activity
{
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
         super.onCreate(savedInstanceState);
         setContentView(R.layout.main);
         TextView text = (TextView)findViewById(R.id.text);
         Typeface font = Typeface.createFromAsset(
           getAssets(), "BaroqueScript.ttf");
         text.setTypeface(font);
    }
}
```

Using Custom Fonts

The visual designer doesn't support custom fonts.