Praktikum Entwicklung Mediensysteme (für Master)

Implementing a User Interface
Outline

• Introduction
• Layouting Components using XML
• Common Layout Objects
• Hooking into a Screen Element
• Listening for UI Notifications
• Applying a Theme to Your Application
Introduction

• Activity
  – Basic functional unit of an Android application
  – But by itself, it does not have any presence on the screen

• Views and Viewgroups
  – Basic units of user interface expression on the Android platform
Views

Implementing a User Interface
Introduction

• `android.view.View`
  - Stores layout and content for a specific rectangular area of the screen
  - Handles measuring and layout, drawing, focus change, scrolling, and key/gestures
  - Base class for widgets
    • `Text`
    • `EditText`
    • `InputMethod`
    • `MovementMethod`
    • `Button`
    • `RadioButton`
    • `Checkbox`
    • `ScrollView`

```java
package com.android.hello;

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 icicle) {
        super.onCreate(icicle);
        TextView tv = new TextView(this);
        tv.setText("Hello, Android");
        setContentView(tv);
    }
}
```
Viewgroups

- android.view.Viewgroup
  - Contains and manages a subordinate set of views and other viewgroups
  - Base class for layouts
Tree-Structured UI

• An Activity in Android
  – Defined using a tree of view and viewgroup nodes

• setContentView() method
  – Called by the Activity to attach the tree to the screen for rendering
LayoutParams

• Every viewgroup class uses a nested class that extends ViewGroup.LayoutParams
  – Contains property types that defines the child’s size and position
Programmatic UI Layout

- Programmatic UI Layout
  - Constructing and building the applications UI directly from source code
  - Disadvantage
    - small changes in layout can have a big effect on the source code

```java
class HelloAndroid extends AppCompatActivity {
  /** Called when the activity is first created. */
  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    TextView tv = new TextView(this);
    tv.setText("Hello, Android");
    setContentView(tv);
  }
}
```
Upgrading UI to XML Layout

• XML-based Layout
  – Inspired by web development model where the presentation of the application’s UI is separated from the logic
  – Two files to edit
    • Java file – application logic
    • XML file – user interface
Upgrading UI to XML Layout
Upgrading UI to XML Layout

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Hello World, UIExample" />
</LinearLayout>
```
UI Example

Hello World, UIExample
Common Layout Objects

Implementing a User Interface
Common Layout Objects

FrameLayout

LinearLayout

TableLayout

AbsoluteLayout

RelativeLayout
FrameLayout

• Simplest layout object

• Intended as a blank reserved space on your screen that you can later fill with a single object
  – Example: a picture that you'll swap out

• All child elements are pinned to the top left corner of the screen

• Cannot specify a location for a child element
Common Layout Objects

FrameLayout

LinearLayout

TableLayout

AbsoluteLayout

RelativeLayout
LinearLayout

- Aligns all children in a single direction — vertically or horizontally
  - All children are stacked one after the other
    - a vertical list will only have one child per row (no matter how wide they are)
    - a horizontal list will only be one row high (the height of the tallest child, plus padding)
Common Layout Objects

- **FrameLayout**
  - Short
  - Button

- **LinearLayout**
  - Longer Button
  - Short

- **TableLayout**
  - Longer Button
  - Short

- **AbsoluteLayout**
  - Longer Button
  - Short

- **RelativeLayout**
  - Longer Button
  - Short
TableLayout

- Positions its children into rows and columns
- Does not display border lines for their rows, columns, or cells
- Cells cannot span columns, as they can in HTML
Common Layout Objects

- FrameLayout
- LinearLayout
- TableLayout
- AbsoluteLayout
- RelativeLayout
Absolute Layout

• Enables children to specify exact x/y coordinates to display on the screen
  – (0,0) is the upper left corner
  – Values increase as you move down or to the right

• Overlapping elements are allowed (although not recommended)

• NOTE:
  – It is generally recommended NOT to use AbsoluteLayout UNLESS you have good reasons to use it
  – It is because it is fairly rigid and does not work well with different device displays
Common Layout Objects

- **FrameLayout**
- **LinearLayout**
- **TableLayout**
- **AbsoluteLayout**
- **RelativeLayout**
RelativeLayout

• Lets children specify their position relative to each other (specified by ID), or to the parent
### Important Layout Parameters

#### Allgemein:

| Layout-Height: | fill_parent, wrap_content, Pixels |
| Layout-Width:  | fill_parent, wrap_content, Pixels |
| Id:            | @+id/my_variable            |
| Min-Height, Max-Height... | |
| Min-Width, Max-Width | |

#### Speziell:

| EditText | Input type | text, textEmailAddress, number, numberDecimal |
| TextView, Button, EditText | Text | @string/resource_id |
| TextView | Text color, Text size | |
Online Reference

Hooking into a Screen Element

Implementing a User Interface
Hooking into a Screen Element
Hooking into a Screen Element

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Hello World, UIExample"
    />
    <EditText
        android:id="@+id/name_entry"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
    />
    <Button
        android:id="@+id/ok"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="OK"
    />
</LinearLayout>
```
Hooking into a Screen Element

@+id syntax:
Creates a resource number in the R class (R.java file) if one doesn't exist, or uses it if it does exist.

Any String value (no spaces)
Hooking into a Screen Element

```java
package pem.samplecode.ui;

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

public class UIExample extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        setContentView(R.layout.main);
        //Add a handle to UI components
        EditText nameEntry = (EditText) findViewById(R.id.name_entry);
        nameEntry.setText("Enter your name here");
    }
}
```
Hooking into a Screen Element
Listening for UI Notifications
Resource Folders and Localization

Implementing a User Interface
Resource Folders
Resource Folders

• Folder structure is automatically parsed into Resource-File

• Do not modify this file!
Resource Folders

• Separate storage of Strings and Graphics
• Makes it easier to modify software parts
• Resources are accessed via „R.java“
Resource Folders
Localization

• Creating folders for other languages does not need any code change

• Watch the application size!
Localization

Strings, with optional simple forr as resources. You can add forma three standard HTML tags: b, i, a or a quote in your string, you mu the whole string in the other kind

Name* hello_world
Value* Hallo, Welt!
Localization

• May be used for other device specific things as well
  – Country
  – Screen dimensions
  – Screen orientation
  – Touchscreen type (finger, stylus)
  – and many more
Application Themes

Implementing a User Interface
Applying a Theme to Your Application

- Default theme: `android.R.style.Theme`

- Two ways to set the theme
  - Adding the theme attribute in `AndroidManifest.xml`
  - Calling `setTheme()` inside the `onCreate()` method
Editing AndroidManifest.xml

• Adding the theme attribute in AndroidManifest.xml
Applying a Theme using Code

• Calling `setTheme()` inside the `onCreate()` method
Black

Light Weight
Exercise 2

Implementing a User Interface
Exercise 2

• Fortführung der bisherigen Aufgabe

• In neues Projekt kopieren

• Browser um Adresszeile und „GoTo“-Button ergänzen

• Zurück und Vorwärts-Button ergänzen

• Vorsicht bei Seiten mit Redirect!
Fragen?
Viel Spaß!