Views

Mildrid Ljosland, Institutt for datateknologi og informatikk (IDI), NTNU
Lærestoffet er utviklet for faget IFUD1042 Applikasjonsutvikling for Android

Resymé: I denne leksjonen tar vi for oss ulike former for views. Vi starter med layout, som er en type view. Ellers ser vi på hvordan man kan få vist fram tekst og bilder og hentet inn opplysninger fra brukeren, samt hvordan man kan lage lyttere. Vi tar også for oss ListView, GridView og Spinner, som alle bruker en Adapter til å putte inn verdier i en liste som vises fram. Tema for denne leksjonen er kapittel 4, 5, 8 og 9. Her er det mange detaljer, du trenger ikke å lære alle i første omgang, du kan gå tilbake og slå opp det du trenger senere.

3.1 Generelt om klassen View

Objekter av klassen View brukes til å lage brukergrensesnittet som en aktivitet består av. Klassen har mange subklasser. Eksempler på dette er TextView, ImageView, ViewGroup, AnalogClock, Button, EditText, diverse layout-klasser og mange flere.

Normalt definerer vi disse komponentene i xml-filer, men det er også mulig å lage dem fra bunnen av i java-klassene. Noen av View-klassene brukes til å vise fram ulike ting, f.eks. tekst eller bilde, mens andre brukes til å organisere komponenter.

Et eksempel på det siste, er klassen ViewGroup. Siden en ViewGroup er en subklasse av View, kan en ViewGroup plasseres alle steder der en View kan plasseres. Det som adskiller en ViewGroup fra en View, er at en ViewGroup kan inneholde en eller flere View-objekter. Dermed kan vi bruke ViewGroup til å bygge opp så kompliserte brukergrensesnitt vi vil, med å legge Views og/eller ViewGroups inni hverandre.

3.2 Layout

Layouter brukes til å organisere de ulike komponentene (viewene) på skjermen. Siden alle layouter er subklasser av ViewGroup, kan vi kan legge layouter inni hverandre.

Den enkleste layouten er LinearLayout, som legger komponentene etter hverandre, enten horisontalt eller vertikalt. Så det enkleste vil være å ha en LinearLayout ytterst, og legge to eller flere andre layouter inni den, og så en eller flere View-objekter av ulike typer inni disse igjen. Det er ikke bestandig like enkelt å få plassert viewene der du ønsker det i layoutene via den grafiske visningen, så du må være forberedt på å jobbe en del i xml-visningen, og flytte elementer der.

AbsoluteLayout er deprecated og bør ikke brukes.

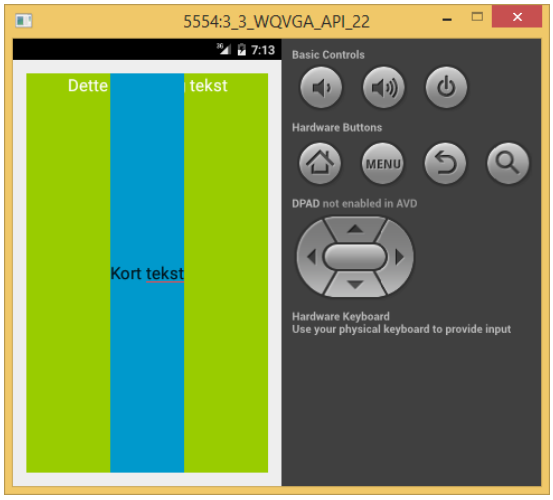
TableLayout legger ut komponentene i en todimensjonal tabell. Hver rekke defineres av en TableRow, og så legger de enkeltkomponentene inni der. Vanligvis har vi like mange kolonner i hver rekke, men det er mulig å ha ulikt antall i de forskjellige rekkene. Da vil det bli like mange kolonner som i den rekka som har flest. I de andre rekkene blir det da tomme plasser på slutten av rekka, eller man kan få en komponent til å strekke seg over flere kolonner ved å bruke android:layout_span.

I RelativeLayout bestemmes plasseringen av de ulike komponentene i forhold til hverandre og i forhold til layouten, ved å spesifisere top, bottom, left, right, above, below osv.

Nedenfor skal vi se på et eksempel som bruker FrameLayout, som legger komponenter oppå hverandre, slik at bare den øverste er synlig hvis den er like stor som de andre. Er den mindre, vil de ytre delene av den neste vises.

Eksempel

Her er et enkelt eksempel på bruk av FrameLayout. Inni FrameLayouten har jeg lagt to tekstbokser, en med lang tekst og en med kort tekst. Siden den lange kommer først, vil den bli plassert nederst (lengst inn i bildet), og tekstboks nummer to oppå. I tillegg til å sette tekstene, har jeg satt bakgrunnsfarger slik at det er tydelig hva som tilhører den ene og den andre. Vi ser på figuren under at den øverste skjuler deler av den nederste.



I utgangspunktet var FrameLayouten definert med størrelse wrap_content på både høyde og vidde, slik at den ikke ble større enn det som krevdes for å få plass til den lengste teksten. Men jeg endret det til fill_parent og fikk den spredt utover hele skjermen. Også for den grønne komponenten (boksen) har jeg satt både høyde og vidde til fill_parent, mens jeg for den blå komponenten bare har satt høyden til fill_parent, mens vidden er beholdt som wrap_content.

I tillegg har jeg brukt attributtene gravity og layout_gravity. Gravity (tyngdekraft) forteller hvordan komponentene skal plasseres, skal den trekkes til høyre eller venstre, opp eller ned? Forskjellen på gravity og layout_gravity er at gravity gjelder det som er inni komponenten, her selve teksten, mens layout_gravity gjelder selve komponenten. For den grønne komponenten har jeg satt gravity til å være center_horisontal. Da havner teksten midt på skjermen horisontalt, men beholder sin vertikale plassering øverst på skjermen. For den blå komponenten har jeg satt layout_gravity til å være center_horisontal, slik at hele boksen plasseres på midten. Og så har jeg satt gravity til center_vertical, slik at teksten flyttes ned til midten av skjermen.

Her er activity_main.xml:

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools=http://schemas.android.com/tools
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    tools:context=".MainActivity">

        <EditText
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:id="@+id/editText"
            android:layout_weight="1"
            android:text="Dette er en lang tekst"
            android:textSize="20dp"
            android:background="@android:color/holo_green_light"
            android:layout_gravity="center"
```

```
            android:gravity="center_horizontal"
            android:textColor="@android:color/white" />
        <EditText
            android:layout_width="wrap_content"
            android:layout_height="fill_parent"
            android:id="@+id/editText2"
            android:layout_weight="1"
            android:text="Kort tekst"
            android:textSize="20dp"
            android:background="@android:color/holo_blue_dark"
            android:layout_gravity="center_horizontal"
            android:gravity="center_vertical" />

</FrameLayout>
```

Mer om layouts finnes på [Layouts].

3.3 Lyttere

Klassen View (og dermed alle subklasser) har metoden onClick(), slik at vi kan definere hva som skal skje når komponenten klikkes. Denne kan enten settes i xml-filen (slik vi gjorde for knappene i forrige leksjon), eller de kan settes i java-programmet. Da må vi lage en lytter, for eksempel slik (forutsetter at vi har laget en knapp med id=button i activity_main.xml):

```
package no.hist.itfag.framelayouttest;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.app.Activity;
import android.widget.Button;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Toast;


public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Button b = (Button)findViewById(R.id.button);
        b.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                Toast.makeText(v.getContext(), "Dette er en test",
                        Toast.LENGTH_LONG).show();
            }

        });
    }
```

OnClickListener er en interface som har metoden onClick. Vi lager altså et objekt av en anonym klasse som implementerer dette interfacet. Dette objektet registeres som lytter for vår komponent ved å kalle setOnClickListener(). Når komponenten klikkes, får lytteren beskjed, og sørger for at onClick-metoden utføres.

Vi kan selvfølgelig også lage en egen (navngitt) lytter-klasse. Den må da implementere interface OnClickListener og ha en metode som heter onClick(View v).

```
public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Button b = (Button)findViewById(R.id.button);
        b.setOnClickListener(new MinLytter());
    }

....

package no.hist.itfag.framelayouttest;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Toast;
public class MinLytter implements OnClickListener {
    public void onClick(View v) {
        Toast.makeText(v.getContext(), "Dette er en ny test",
                Toast.LENGTH_LONG).show();
    }
}
```

Eller vi kan Activity-klassen implementere OnClickListener, og la onClick() være en metode i den klassen:

```
public class MainActivity extends Activity implements OnClickListener {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Button b = (Button)findViewById(R.id.button);
        b.setOnClickListener(this);
    }
    @Override
    public void onClick(View v) {
        Toast.makeText(v.getContext(), "Dette er en annen test",
                Toast.LENGTH_LONG).show();
    }
```

Her er this argumentet til setOnClickListener, siden this er Activity-objektet, og det er dette objektet sin onClick() vi skal bruke.

Det finnes også andre lyttere, for eksempel OnFocusChangeListener og OnKeyListener med metodene henholdsvis OnFocusChange() og OnKey(). I eksempelet i slutten av leksjonen bruker vi diverse lyttere. Du kan lese mer om lyttere i [Input Events]

3.4 Views som trenger en adapter

ListView, SpinnerView og GridView lar oss vise fram lengre lister som bruker kan velge fra. Hovedforskjellen på dem er at ListView og GridView viser hele lista, mens Spinner bare viser ett av elementene. Men hvis vi klikker på den, kommer hele lista fram. ListView har bare en kolonne, mens GridView kan ha flere.

Merk at det finnes en ferdigdefinert klasse ListActivity som har som standard en ListView (istedenfor det vanlige TextView). Denne klassen kan vi lage en subklasse av. Da vil onCreate()-metoden bestå i å fylle lista, mens den vanlige setContentView kan utkommenteres. Det er selvfølgelig også mulig å putte en ListView inn i en vanlig layout, på samme måte som alle andre Views.

For alle tre klassene trenger vi et objekt av klassen Adapter for å fylle lista. Det finnes mange typer adaptere, men den vanligste er ArrayAdapter<String>. I en slik kan vi fylle streng-data fra en tabell, for eksempel slik:

```
ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
        android.R.layout.simple_spinner_item, dyrenavn);
```

Hvis du vil vite mer om adaptere, kan du for eksempel se på [Adapter].

I android.R.layout finnes mange forskjellige layout-varianter som styrer utseende av lista, for eksempel kan vi lage en radioknappliste ved å skrive simple_list_item_choice. Eksperimenter gjerne med dem. Den tredje parameteren er den tabellen vi skal putte inn. Når vi har fått dataene inn i adapteren, putter vi adapteren inn i lista:

```
spinner.setAdapter(adapter);
```

eller

```
this.setListAdapter(adapter);//Hvis vi bruker ListActivity
```

Og så må vi selvfølgelig lage og registere en onClick (eller tilsvarende)-metode som sørger for å utføre brukerens valg:

```
spinner.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> parent,
                  View valgt, int posisjon, long id) {
                  //Et eller annet
            }
            @Override
            public void onNothingSelected(AdapterView<?> arg0) {
                  // TODO Auto-generated method stub
            }
});
```

Eventuelt

```
list.setOnItemClickListener(new AdapterView.OnItemClickListener() {

      @Override
      public void onItemClick(AdapterView<?> parent, View valgt,
            int posisjon, long id) {
            // Et eller annet
      }

});
```

(Skrivemåten <?> i AdapterView forteller oss at AdapterView er en generisk type, men vi ønsker ikke å spesifisere en bestemt type i metode-definisjonen. Se mer om dette på [WikipediaGenerics])

3.5 Eksempel

Vi skal lage en applikasjon som omhandler dyr. Vi ønsker både å kunne vise beskrivelser av ulike dyr, samt bilder av dem. Poenget med dette eksemplet er å vise ulike måter det kan gjøres på, så det inneholder ulike komponenter som gjør mye av det samme.

Først lager jeg et nytt prosjekt med navnet Dyr. Jeg starter med å laste ned fem dyrebilder fra [Iconspedia], både i liten (32x32) og stor (128x128) versjon og lagre dem i drawable-katalogen. Bildene får navnene dyr1.png, dyr1_icon.png, dyr2.png og så videre.

Deretter lagret jeg dyrenavnene og beskrivelser av dyrene som strenger i strings.xml :

```
<resources>
<string name="app_name">Dyr</string>
<string name="hello_world">Hello world!</string>
<string name="action_settings">Settings</string>
<string name="avslutt">Avslutt</string>
<string name="dyr1">Elefant</string>
<string name="dyr2">Giraff</string>
<string name="dyr3">Løve</string>
<string name="dyr4">Flodhest</string>
<string name="dyr5">Sebra</string>
<string name="dyrb1">Stor og tung. Har lang snabel og store ører. Kan
brukes til  ridedyr og arbeidshest.</string>
<string name="dyrb2">Lang hals og lange bein.</string>
<string name="dyrb3">Farlig. Spiser andre dyr.</string>
<string name="dyrb4">Stor og lever i vann.</string>
<string name="dyrb5">Svart- og hvit-stripet hest.</string>
```

Til slutt lager jeg tabeller av disse verdiene ved å putte inn

```
<string-array name="dyrenavn">
    <item>@string/dyr1</item>
    <item>@string/dyr2</item>
    <item>@string/dyr3</item>
    <item>@string/dyr4</item>
    <item>@string/dyr5</item>
</string-array>
<string-array name="dyrebeskrivelse">
    <item>@string/dyrb1</item>
    <item>@string/dyrb2</item>
    <item>@string/dyrb3</item>
    <item>@string/dyrb4</item>
    <item>@string/dyrb5</item>
</string-array>
<array name="bilder">
    <item>@drawable/dyr1 </item>
    <item>@drawable/dyr2</item>
    <item>@drawable/dyr3</item>
    <item>@drawable/dyr4</item>
    <item>@drawable/dyr5</item>
</array>
<array name="ikoner">
    <item>@drawable/dyr1_icon</item>
    <item>@drawable/dyr2_icon</item>
    <item>@drawable/dyr3_icon</item>
    <item>@drawable/dyr4_icon</item>
    <item>@drawable/dyr5_icon</item>
</array>

</resources>
```

Her har jeg definert fire tabeller, en med dyrenavnene, en med beskrivelsene og to med bildene. Strenger lagres i string-array, andre typer data lagres i array.

Så begynner jeg på layouten. I activity_main.xml legger jeg først inn en Spinner (id:spinnerDyr) og en TextView (id:beskrivelse). Tanken er at beskrivelsen av det dyret som er valgt i spinneren, skal vises i tekstboksen. Deretter legger jeg inn en LinearLayout med horisontal orientering. Her skal jeg senere legge i bildeknapper, dette kommer jeg tilbake til. Et trykk på en av disse knappene skal føre til at det store bildet av dyret vises, mens et trykk på det store bildet sender oss tilbake til hovedskjermbildet. Så legger jeg inn en ListView-komponent og en GridView-komponent inni en ny LinearLayout som også er horisontalt orientert, slik at de kommer ved siden av hverandre. ListView-komponenten skal fungere på samme måte som spinneren, mens GridView-komponenten skal fungere som knappene. Til slutt legger jeg inn en vanlig knapp (id: buttonAvslutt) som kan brukes til å avslutte aktiviteten. Den blir plassert i bunnen. Nedenfor vises hvordan det ser ut når det er ferdig.

Layouten ser nå slik ut:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools=http://schemas.android.com/tools
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    tools:context=".MainActivity">

    <Spinner
        android:id="@+id/spinnerDyr"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"/>

    <TextView
        android:id="@+id/beskrivelse"
        android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_below="@+id/spinnerDyr"
        />

    <LinearLayout
        android:id="@+id/linearLayout1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_below="@+id/beskrivelse"
        android:orientation="horizontal"
        />
    <LinearLayout
        android:id="@+id/linearLayout2"
        android:layout_below="@+id/linearLayout1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">
    <ListView
        android:id="@+id/listView1"
        android:layout_weight="1"
        android:layout_width="0dp"
        android:layout_height="match_parent"/>
    <GridView
        android:id="@+id/gridView1"
        android:layout_weight="2"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:numColumns="auto_fit" >
    </GridView>
    </LinearLayout>
    <Button
        android:id="@+id/buttonAvslutt"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:text="@string/avslutt" />

</RelativeLayout>
```

I ListView er layout_weight satt til 1, mens den i GridView er satt til 2. Det fører til at GridView-komponenten vil ta dobbelt så stor plass som ListView-komponenten.

Jeg lager også en annen layout-fil som jeg kaller vis_dyr.xml. Der har jeg en ImageView (id:stortBilde), og setter layout_height og layout_width til match_parent for begge. Denne layouten skal brukes til å vise fram de store bildene.

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <ImageView
        android:id="@+id/stortBilde"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_alignParentRight="true"
        android:layout_centerVertical="true"
```

```
        android:layout_marginRight="66dp"
        android:src="@drawable/dyr1_icon" />
</RelativeLayout>
```

Nå er neste punkt å implementere funksjonaliteten i klassen MainActivity.

Det første jeg gjør, er å hente inn tabellene med dyrenavn, beskrivelser og bilder:

```
private String[] dyrenavn;
private String[] dyrebeskrivelse;
private TypedArray ikoner;
private TypedArray bilder;
private int[] imageIDs;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    Resources res = getResources();
    dyrenavn = res.getStringArray(R.array.dyrenavn);
    dyrebeskrivelse = res.getStringArray(R.array.dyrebeskrivelse);
    ikoner = res.obtainTypedArray(R.array.ikoner);
    bilder = res.obtainTypedArray(R.array.bilder);
```

(Det som i strings.xml er spesifisert som string-array, blir til String[], mens array blir til en spesiell tabelltype som kalles TypedArray.)

Ved å legge inn alle strengene og bildene i tabeller og definere dem i strings.xml, blir programkoden helt uavhengig av hvor mange dyr jeg har. Hvis jeg på et senere tidspunkt vil legge inn enda et dyr, må jeg oppdatere strings.xml, samt legge inn png-bildene, men jeg trenger ikke å endre programkoden, og heller ikke layout-fila.

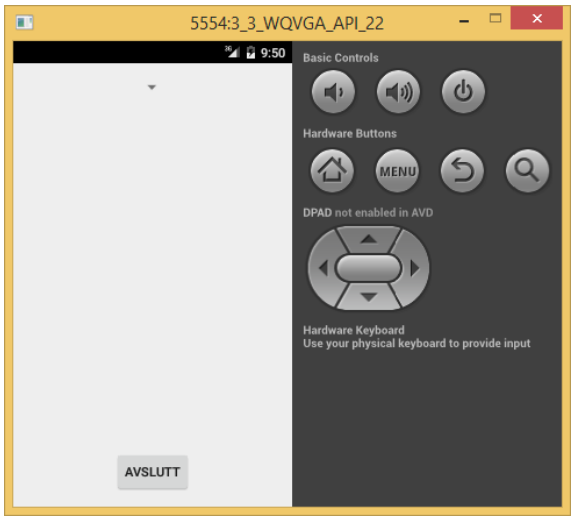
Knapper

Så skal jeg initiere de ulike komponentene. Jeg starter med det enkleste, avslutt-knappen:

```
    private void initerAvsluttknapp() {
          Button avslutt = (Button)findViewById(R.id.buttonAvslutt);
            avslutt.setOnClickListener(new View.OnClickListener(){
                  public void onClick(View v) {
                        DyrActivity.this.finish();
                  }
            });
   }
```

Det eneste jeg trnger å gjøre, er å lage en lytter som sørger for å avslutte aktiviteten. findViewById() kan brukes på alle View, det krever bare at komponenten er definert med en id=… Button-komponenten er definert med **android:id="@+id/buttonAvslutt**", så da kan vi referere til den ved R.id.buttonAvslutt. Returtypen til findViewById er View, men vi «caster» til riktig type (her Button) slik at vi kan bruke de metodene denne typen har.

Ved å legge inn kallet initierAvsluttknapp(); nederst i onCreate(), kan programmet kjøres og vi får følgende skjermbilde (pilen øverst er spinneren uten noen elementer):



Så var det bildeknappene. Her skal jeg lage knappene direkte i programmet slik at jeg kan lage nøyaktig så mange jeg trenger, avhengig av antall bilder jeg har. Først leter jeg fram den LineærLayouten jeg laget, og definerer hvilke layout-parametre jeg vil ha på knappene. Så går jeg i løkke og lager en og en knapp.

Jeg går gjennom hele bilde-tabellen og lager den tilsvarende knappen. Så setter jeg hvilket bilde den skal vise fram og hvilke parametre den skal ha samt legger den inn i layouten. Til slutt setter jeg en lytter på den.

```
private void initierKnapper() {
    LinearLayout layout = (LinearLayout)findViewById(R.id.linearLayout1);
    LayoutParams params =
            new LinearLayout.LayoutParams(
                    LayoutParams.WRAP_CONTENT, LayoutParams.WRAP_CONTENT);
    for (int i=0; i < ikoner.length(); i++) {
        ImageButton knapp = new ImageButton(this);
        knapp.setImageDrawable(ikoner.getDrawable(i));
        knapp.setLayoutParams(params);
        layout.addView(knapp);

        final int nr = i; // Må være final for å brukes i indre klasse
        knapp.setOnClickListener(new View.OnClickListener(){
            public void onClick(View v) {
                visStortBilde(nr);
            }
        });
    }
}
```

Lytterens onClick-metoden kaller metoden visStortBilde. Her er denne metoden:

```
    private void visStortBilde(int nr) {
      Intent i = new Intent("no.hist.itfag.dyr.VisBilde");
      i.putExtra("nr", nr);
      startActivity(i);
    }
```

Jeg lager altså en ny intent, sender med hvilket dyr det er snakk om, og starter aktiviteten VisBilde.

For at dette skal fungere, må jeg lage aktiviteten VisBilde. Den ser slik ut:

```
package no.hist.itfag.dyr;
import android.os.Bundle;
import android.app.Activity;
import android.content.Intent;
import android.widget.ImageView;
import android.content.res.TypedArray;
import android.graphics.drawable.Drawable;
import android.view.View;

public class VisBilde extends Activity {
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.vis_dyr);
        settLytter();
        Intent i = getIntent();
        int nr=i.getIntExtra("nr", 0);
        visValgtDyr(nr);
    }
    private void visValgtDyr(int nr) {
        if (nr >= 0) {
            ImageView bildeView = (ImageView)findViewById(
                      R.id.stortBilde);
            TypedArray bilder = getResources().obtainTypedArray(
                    R.array.bilder);
            Drawable bilde = bilder.getDrawable(nr);
            bildeView.setImageDrawable(bilde);
        }
    }
    private void settLytter() {
        ImageView bildeView = (ImageView)findViewById(R.id.stortBilde);
        bildeView.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                VisBilde.this.finish();
            }
        });
    }
}
```

Og her er tilhørende xml-fil:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <ImageView
        android:id="@+id/stortBilde"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_alignParentRight="true"
        android:layout_centerVertical="true"
        android:layout_marginRight="66dp"
        android:src="@drawable/dyr1_icon" />
</RelativeLayout>
```

Vi har altså et ImageView som heter «stortBilde». Inni der skal vi putte et dyrebilde. Hvilket som skal vises fram, hentes ut fra Intenten ved å bruke getIntExtra(). Vi henter ut riktig bilde, lager en Drawable av det og putter det inn i ImageViewen.

Jeg har også lagt inn en lytter i denne aktiviteten. Den sørger for å sende oss tilbake til hovedaktiviteten igjen når vi klikker på bildet.

Siden vi nå har en ekstra aktivitet, må den inn i manifestfila, så den ser nå slik ut:

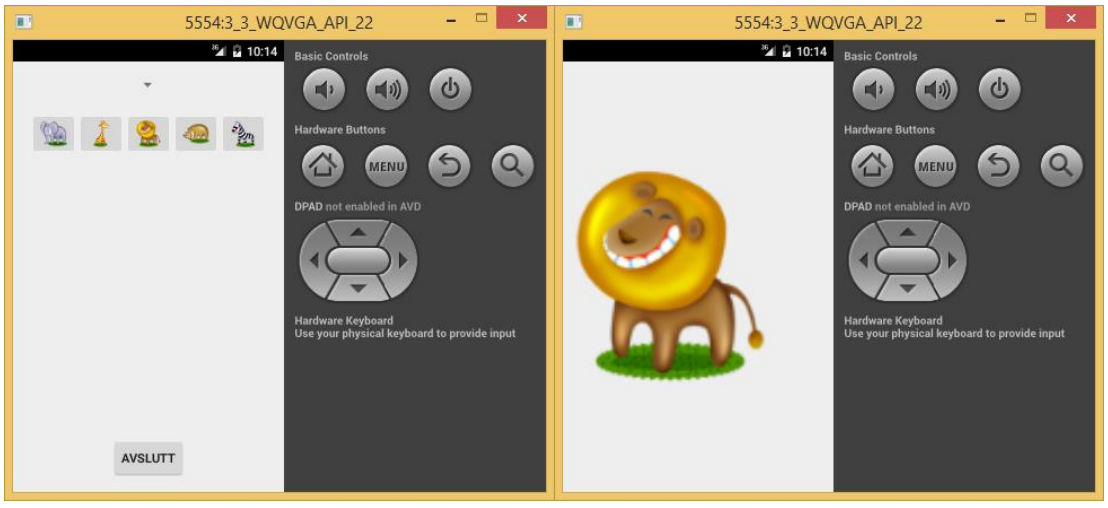
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="no.hist.itfag.dyr" >

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER"
/>
            </intent-filter>
        </activity>
        <activity
            android:name=".VisBilde"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="no.hist.itfag.dyr.VisBilde" />
                <category android:name="android.intent.category.DEFAULT" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Da kan jeg legge inn et kall til initierKnapper() i onCreate() for MainActivity, og får følgende skjermbilde (før og etter vi har klikket på løvebildet):



Lister

Vi har tre forskjellige lister som alle bruker adapterklassen. I ListView og Spinner bruker eg ArrayAdapter<String> mens jeg for GridView har laget en egen adapter som er en subklasse av BasicAdapter.

Konstruktøren til ArrayAdapter har tre parametre, context, layout og array. Context vil normalt være this, mens layout kan være en av standard-layoutene som finnes i Android eller det kan være en egendefinert layout. Arry er den lista som skal puttes inn.

Først ser vi på ListView-komponenten. initierList() ser slik ut:

```
private void initierList() {
      ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
                android.R.layout.simple_list_item_activated_1, dyrenavn);
      ListView listView = (ListView)(findViewById(R.id.listView1));
      listView.setAdapter(adapter);
      listView.setChoiceMode(ListView.CHOICE_MODE_SINGLE);
      listView.setOnItemClickListener(new AdapterView.OnItemClickListener()
      {
          @Override
          public void onItemClick(AdapterView<?> parent,
                                  View valgt, int posisjon, long id) {
              TextView tv = (TextView) findViewById(R.id.beskrivelse);
              tv.setText(dyrebeskrivelse[posisjon]);
          }
      });
```

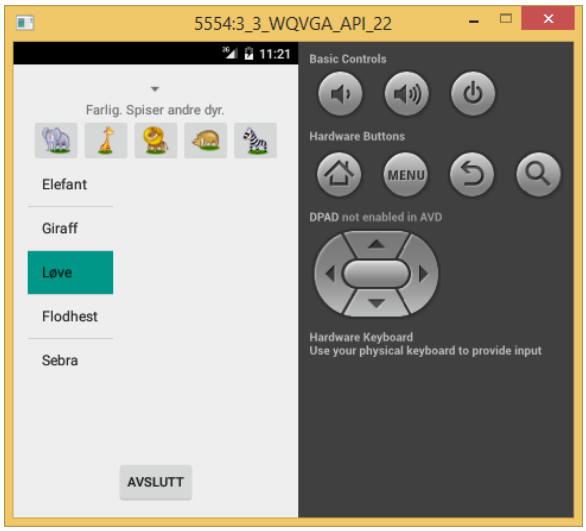
I adapteren bruker jeg layouten android.R.layout.simple_list_item_activated_1, altså en av Android sine ferdigdefienerte layouter. Som navnet sier, er det en enkel layout. Men activated forteller at den markerer det elementet vi klikker på. For å få dette til å fungere, må jeg gi beskjed om at bare ett element kan aktiveres om gangen, ved å skrive

listView.setChoiceMode(ListView.***CHOICE_MODE_SINGLE***);

Lista som puttes inn, er lista med dyrenavn, altså elefant, giraff og så videre.

Også her setter jeg på en lytter. Men her heter den ikke bare onClickListener, men onItemClickListener, og metoden heter onItemClick(). Det som skjer når vi klikker på et av elementene, er at den tilhørende beskrivelsen vises fram.

Ved å putte inn kall til initierList() i onCreate(), og kjøre, vises følgende (før og etter vi har klikket på løve-elementet):



Vi ser at løve-elementet er blitt markert, og teksten «Farlig. Spiser andre dyr.» kommer fram.

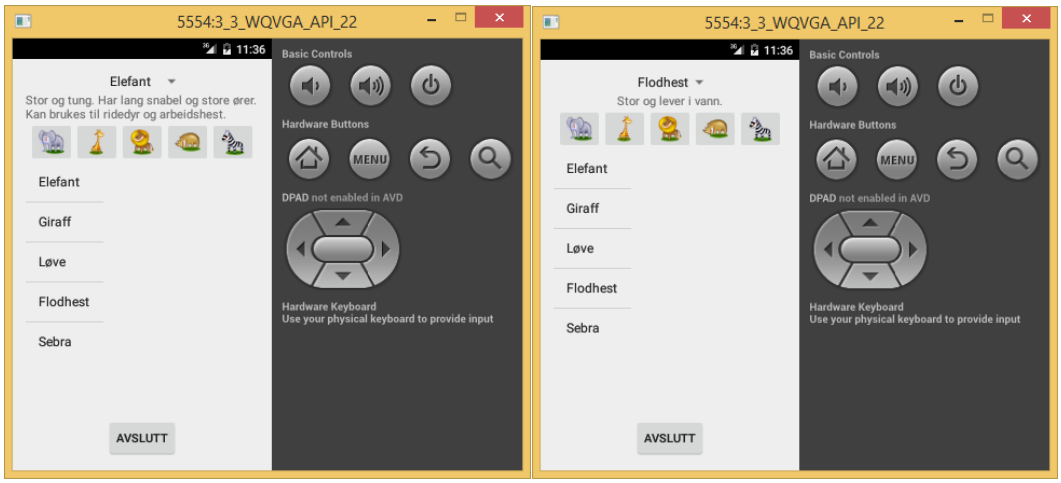
Spinneren lages på mye den samme måten.

```
private void initierSpinner() {
    ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
            android.R.layout.simple_spinner_item, dyrenavn);
    Spinner spinner = (Spinner)findViewById(R.id.spinnerDyr);
    spinner.setAdapter(adapter);
    spinner.setOnItemSelectedListener(new
                AdapterView.OnItemSelectedListener() {
        @Override
        public void onItemSelected(AdapterView<?> parent,
                                   View valgt, int posisjon, long id) {
            TextView tv = (TextView) findViewById(R.id.beskrivelse);
            tv.setText(dyrebeskrivelse[posisjon]);
        }

        @Override
        public void onNothingSelected(AdapterView<?> arg0) {
            // TODO Auto-generated method stub
        }
    });
}
```

Jeg setter adapteren til å inneholde dyrenavnene, og bruker den ferdigdefinerte `simple_spinner_item` til å vise dem fram. Nå heter lytteren OnItemSelectedListener, og finnes som en indre klasse i AdapterView. I onItemSelected henter jeg ut det valgte dyret og setter beskrivelsen som tekst i tekstboksen. Metoden onNothingSelected()kreves implementert, men siden jeg ikke trenger den her, lar jeg den være tom.

Så da får vi følgende skjerbilde:



Både ListView-komponenten og spinneren bruker det samme tekstfeltet til å skrive ut beskrivelsen. Hvis vi ønsker å binde dem sammen, slik at spinneren også viser det som ble valgt i listviewet, kan vi i initerList() sin onItemClick() utkommentere de to linjene som endrer textviewet, og i stedet skrive

```
Spinner spinner = (Spinner)findViewById(R.id.spinnerDyr);
spinner.setSelection(posisjon);
```

Da vil spinneren få samme posisjon som listviewet, og spinnerens lytter sørger for å vise fram riktig beskrivelse.

Til slutt har vi GridView-komponenten. Vi starter med å hente fram id'ene til bildene. Bilder er en TypedArray som har metoden peekValue().resourceId for å finne selve bildet på den gitte indeksen.

Lytteren sørger for å vise fram det store bildet (akkurat som bildeknappene), mens adapterklassen er egendefinert.

Programkoden for selve komponenten blir dermed:

```
private void initierGrid() {
    imageIDs=new int[bilder.length()];
    for (int i=0; i < bilder.length(); i++) {
        imageIDs[i]=bilder.peekValue(i).resourceId;
    }
    GridView g = (GridView)findViewById(R.id.gridView1);
    g.setAdapter(new MinAdapter(this));
    g.setOnItemClickListener(new OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> parent, View valgt,
                        int posisjon, long id) {
            visStortBilde(posisjon);
        }
    });
}
```

For å lage en egendefinert adapterklasse, må vi implementere diverse metoder. Her er alt gjort på enkleste måte, og bare getView() omtales nedenfor. Klassen MinAdapter er laget som en indre klasse til MainActivity.

```
public class MinAdapter extends BaseAdapter {
    private Context context;

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

    @Override
    public int getCount() {
        return imageIDs.length;
    }
```

```
    @Override
    public Object getItem(int position) {
        return imageIDs[position];
    }

    @Override
    public long getItemId(int position) {
        return position;
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        ImageView imageView;
        if (convertView == null) {
            imageView = new ImageView(context);
            imageView.setImageResource(imageIDs[position]);
            imageView.setLayoutParams(new GridView.LayoutParams(50,50));
        } else {
            imageView = (ImageView)convertView;
        }
        return imageView;
    }
}
```

I getView() lager vi et ImageView, slik at bildene vises fram. Hvis viewet ikke finnes fra før, lager vi et nytt ved å hente fram bildet i riktig posisjon, og sette størrelse på det. Her har jeg hardkodet størrelsen til 50x50, men hvis jeg ville gjøre det mer fleksibelt, kunne disse verdiene vært satt i dimens.xml.

I activity_main.xml er numColums satt til auto_fit, slik at systemet selv finner ut hvor mange kolonner som er mest hensiktsmessig. Jeg kunne alternativt ha satt det til 1, slik at bildene kom under hverandre, eller f.eks. 3, slik at det kom tre bilder på hver linje.

Dette eksemplet finner du som zip-fil sammen med resten av leksjonen.

3.6 Referanser

[Layouts] http://developer.android.com/guide/topics/ui/declaring-layout.html

[Input Events] http://developer.android.com/guide/topics/ui/ui-events.html

[Adapter] http://developer.android.com/reference/android/widget/Adapter.html

[WikipediaGenerics] http://en.wikipedia.org/wiki/Generics_in_Java#Wildcards

[Iconspedia] http://www.iconspedia.com