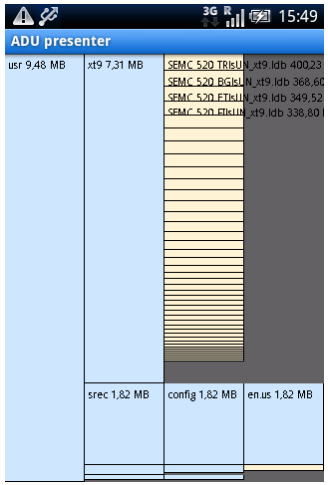
Forside til eksamensprojekt

INGENIØR
HØJSKOLEN
KØBENHAVN

Center for Videreuddannelse
Lautrupvang 15, Indgang 19 B
DK- 2750 Ballerup
Tlf: 44 80 51 00
Fax: 44 80 51 10
Mail: cv@ihk.dk

	OOPA/MGA-T-401									

Sæt X i rubrikken under dit fag:



Underviser: ___Jacob Nordfalk_____

Eksamensform: Eksamensform kan ej vælges - er udelukkende individuel eksamen

Enmandsprojekt: ☒ Gruppeprojekt ☐

Studienr: ____101800____ Navn: _____Lise Andreasen_____

Studienr: _____ Navn: _____

Titel_____Android Disk Usage (ADU)_____

Fortroligt ☐

Må anvendes i undervisningen X

Dato: __9____-____12____-_____2010_____

_____ Den studerendes underskrift	_____ Den studerendes underskrift – kun ved gruppeprojekt

Afleveringsfrist:

Projektet skal afleveres i 3 eksemplarer senest torsdag den 9. december 2010, mellem kl. 09.00 og kl. 12.00 i sekretariatet på CV, Lautrupvang, 19b, 1. sal, 2750 Ballerup
Projekter modtages **IKKE** i ringbind eller brevordnere - kun i mindre tilbudsmapper eller med spiralryg af hensyn til videreforsendelse til vejleder og censor.

Indholdsfortegnelse

Android Disk Usage (ADU)

Indledning / forord

Blandt UNIX/Linux-folk er `du` en elsket kommando. Den giver hurtigt overblik over, hvor meget af disk-systemet, der allerede er brugt, og hvad der for alvor fylder. Der er grund til at tro, at Android-folk kunne have tilsvarende behov (omend svaret jo nok altid vil være: det er mp3'er og videoer, der tager al pladsen).

Problemformulering

> Skab et disk usage program, der passer til Android-miljøet, og kører korrekt.

Nærmere bestemt: portér JDU[1], der er skabt i Java og kører i vindues-miljøer.

Da JDU bl.a. benytter sig af mus, og har sine kommandoer fordelt i samme vindue, bliver en del af processen også at omskabe programmet til Android, der er præget af at have forskellige skærmbilleder til forskellige kommandoer. En fil-browser[2] bliver tilføjet, så den rigtige mappe kan findes, og siden tilføjes en hjælper[3], der kan lave en lidt pænere liste over mapper.

Afgrænsning

Fra starten af afgrænses to dele af JDU: at vise teksten i varierende størrelse, afhængig af mappens/filens størrelse, og at opdatere en progressbar, der giver vældig god mening i vindues-miljøer, men i Android enten skal droppes eller erstattes af Androids ”jeg arbejder” mekanisme.

Den oprindelige JDU blev først kraftigt beskåret, og siden blev et par klasser omdøbt. Således er `JDUDirectory.java`, `JDUFile.java` og `PregressObserver.java` stort set urørte i forhold til den oprindelige JDU, mens en del arbejde er foregået i `TableView.java`, der viser resultatet af at gennemsøge filsystemet.

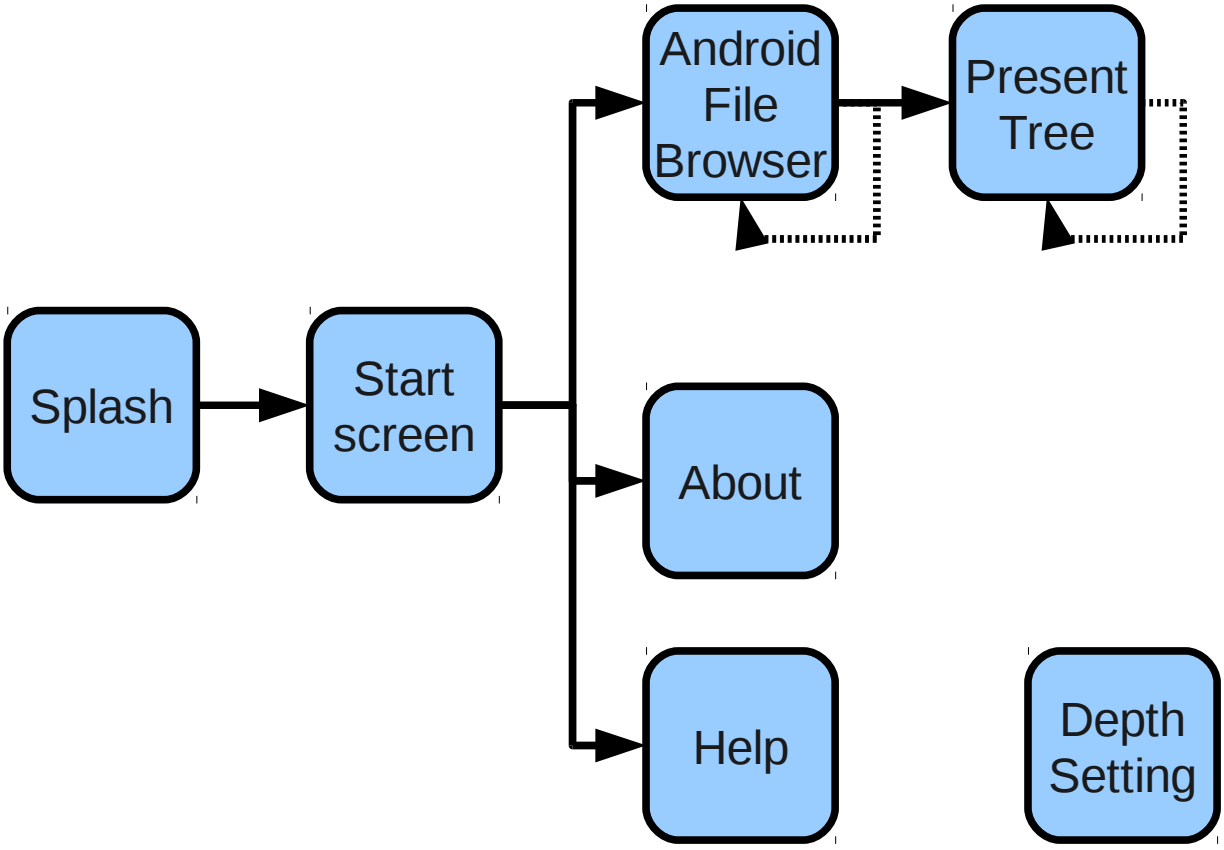
`AfbAdapter.java` og `AfbOption.java` er omdøbt og svagt modificeret.

1 http://jasonpratt.org/software/jdu/
2 http://www.anddev.org/viewtopic.php?t=67
3 http://trace.adityalesmana.com/2010/08/customize-android-listview-via-listadapter/

Selve analysen

Diagrammer

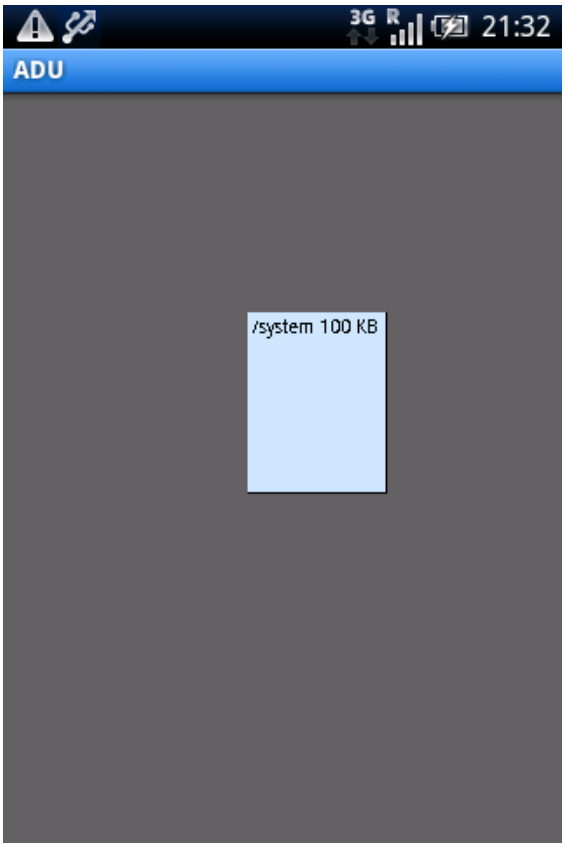
For at få lidt styr på, hvilket program jeg gerne vil lave, starter jeg med et diagram over de anvendte activities. Ud af dette kan jeg også regne ud, hvilke klasser jeg skal have i gang, der ligger bag den enkelte aktivitet. Samtidig kan jeg begynde at planlægge layouts og view.



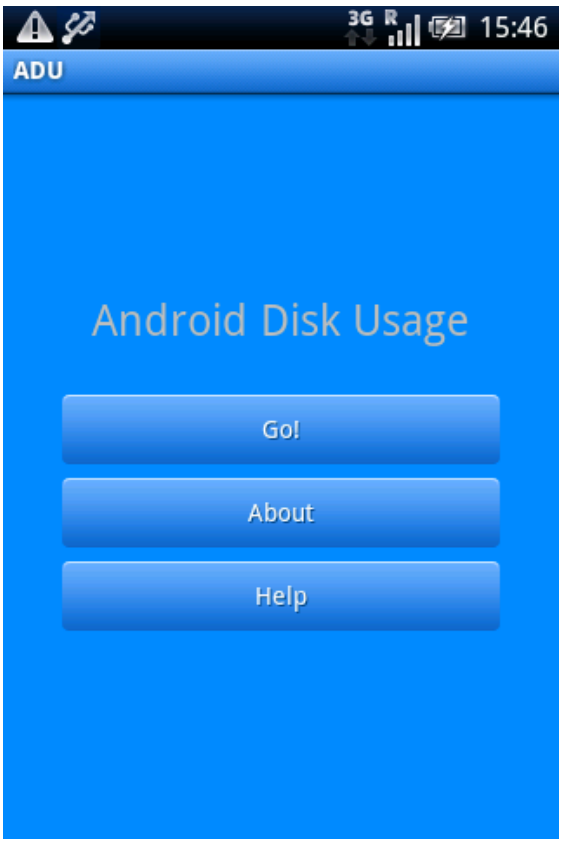
`AndroidFileBrowser` har i sig et view, der kan erstattes af et nyt view. `PresentTree` er i samme situation. Derfor de 2 stiplede linjer. DepthSetting er en menu-aktivitet.

Herefter planlægger jeg mine skærmbilleder. I det følgende vises skærmbilleder fra en version af programmet, det færdige program ser muligvis anderledes ud.

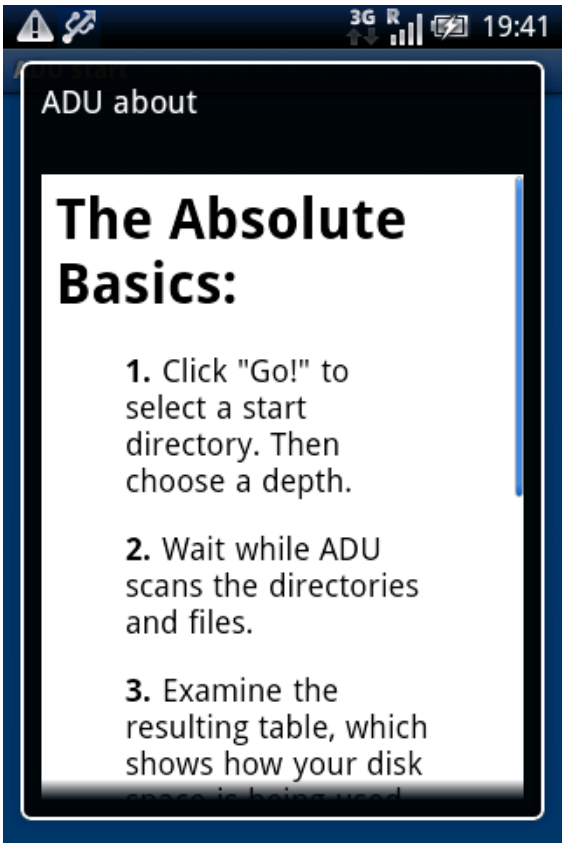
Uafhængigt af hvor man ellers er i programmet kan man vælge, hvilken dybde filsystemet skal vises med.



Før vi går i gang, er der en splash-skærm. En enkelt mappe løber rundt på skærmen, afhængig af hvordan telefonen vender. Ved et enkelt tap kommer man videre til den egentlige startskærm.



Startskærmen skal have 3 knapper, der fører videre til hver sin nye aktivitet. Baggrundsfarven er tilpasset den blå, der senere bruges i præsentation af træet.

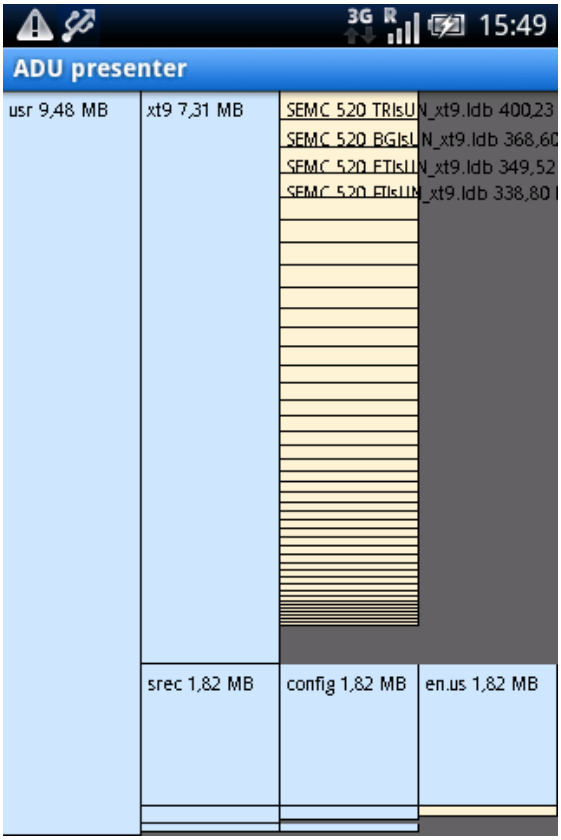


Hjælpeskærmen træder ind foran startskærmen, og fortæller lidt om, hvordan programmet virker. Man bakker for at komme tilbage til start.

Tilsvarende fortæller About lidt om hvem der har lavet programmet.

Nu er vi i gang. Vi bliver præsenteret for en liste af de mapper, der ligger i det øverste niveau af filsystemet. For hver mappe er der to muligheder: analysér denne mappe, og gå ned i denne mappe. Sidste mulighed bruges, når den mappe, der skal analyseres, ligger længere nede.

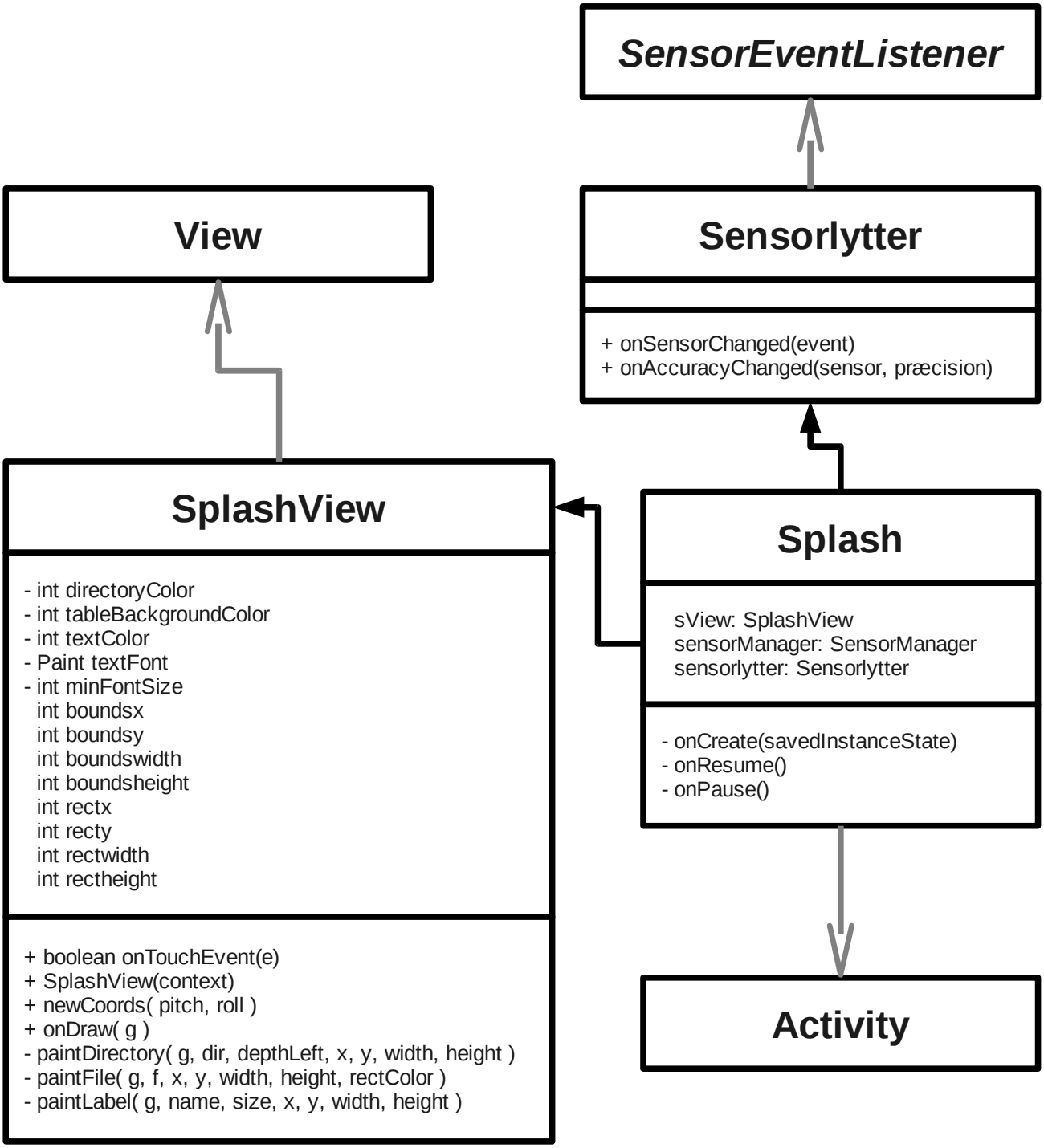
Her er der tappet på ”Enter /system”, og den valgmulighed lyser op. Denne aktivitet genstarter nu, i /system i stedet for roden. Hvis der var blevet valgt ”Analyze”, var vi gået videre til næste del.



Her er vi i sidste trin, hvor usr er valgt som den øverste mappe, der skal analyseres, og i alt 4 niveauer vises. Hvis man fx tapper på config, bliver den den mappe, der vises yderste til venstre, og strukturen gentegnes. Hvis man er gået ned i træet, kan man komme op igen ved at tappe på den venstre mappe.

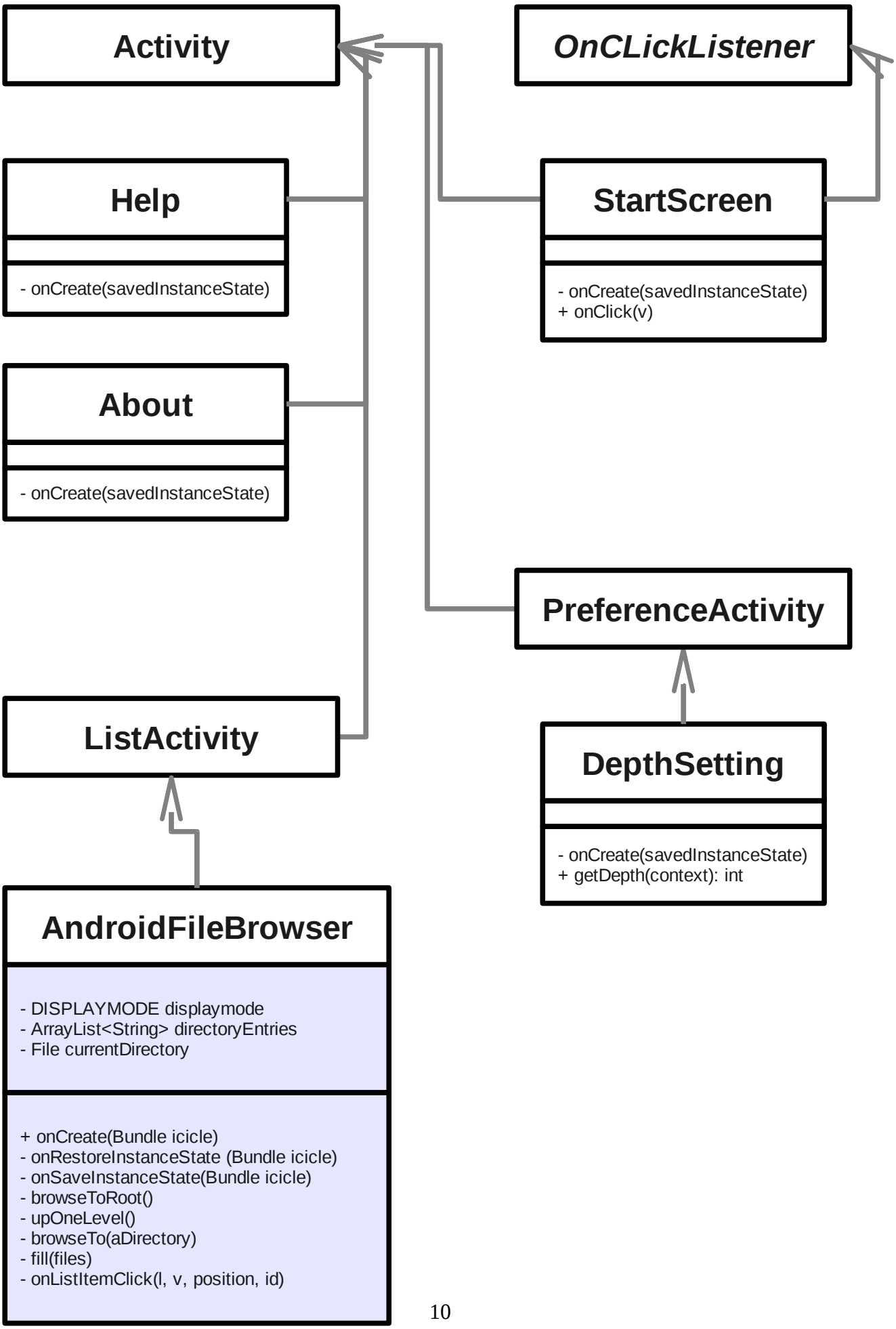
Efter at have planlagt aktiviteter og skærmbilleder, går jeg i gang med klasser. Nedenstående skemaer er igen dannet efter programmeringen var gået i gang, og visse detaljer kan have ændret sig

inden det færdige program forelå.



For at starte godt, har vi en `Splash`. Udover at være en underklasse til `Activity`, har den også en `SensorEventListener`, der lægger mærke til hvordan telefonen vender.

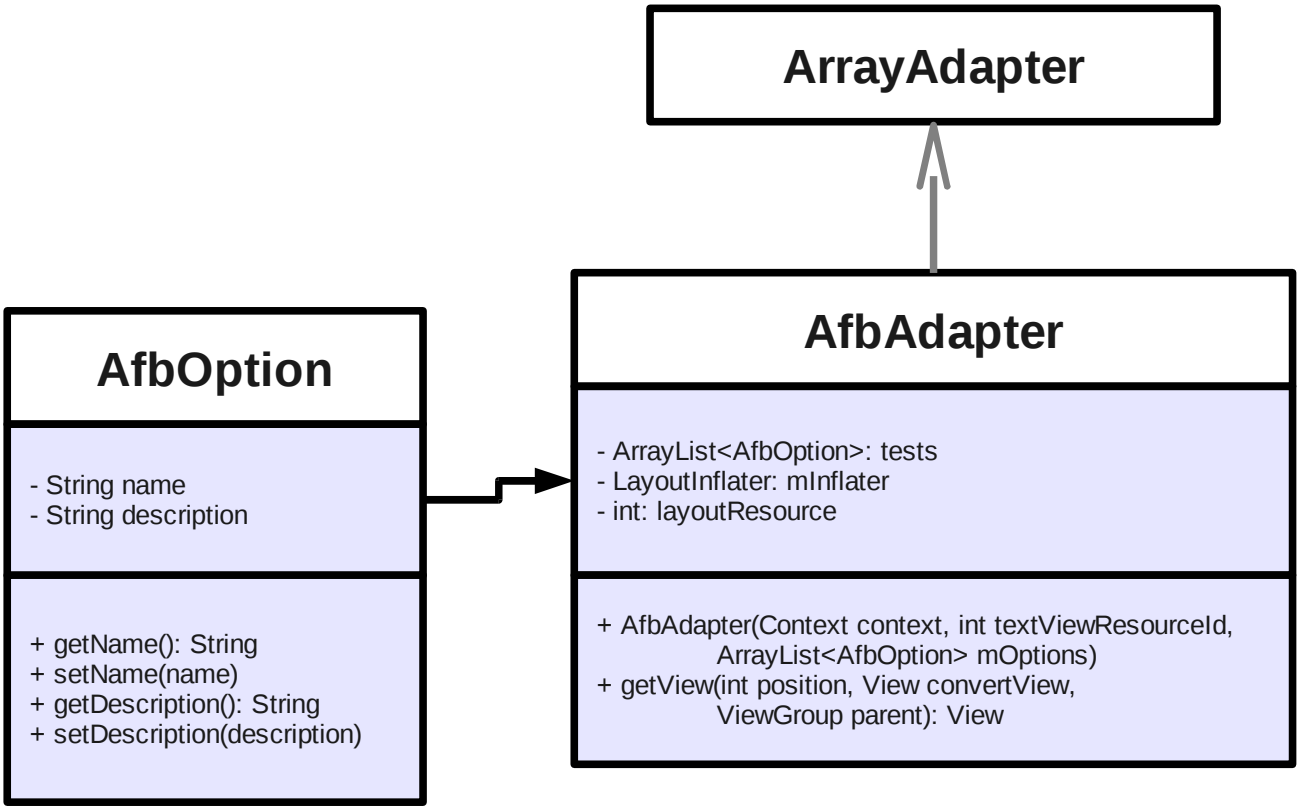
For at få tegnet noget på skærmen, er der også brug for `SplashView`, en underklasse til `View`. Den er meget i familie med `TableView`, og med lidt forsigtighed kunne der være noget med nedarvning fra en fælles overklasse.


Activity
OnCLickListener
Help
- onCreate(savedInstanceState)
StartScreen
- onCreate(savedInstanceState)
+ onClick(v)
About
- onCreate(savedInstanceState)
PreferenceActivity
ListActivity
DepthSetting
- onCreate(savedInstanceState)
+ getDepth(context): int
AndroidFileBrowser
- DISPLAYMODE displaymode
- ArrayList<String> directoryEntries
- File currentDirectory
+ onCreate(Bundle icicle)
- onRestoreInstanceState (Bundle icicle)
- onSaveInstanceState(Bundle icicle)
- browseToRoot()
- upOneLevel()
- browseTo(aDirectory)
- fill(files)
- onListItemClick(l, v, position, id)

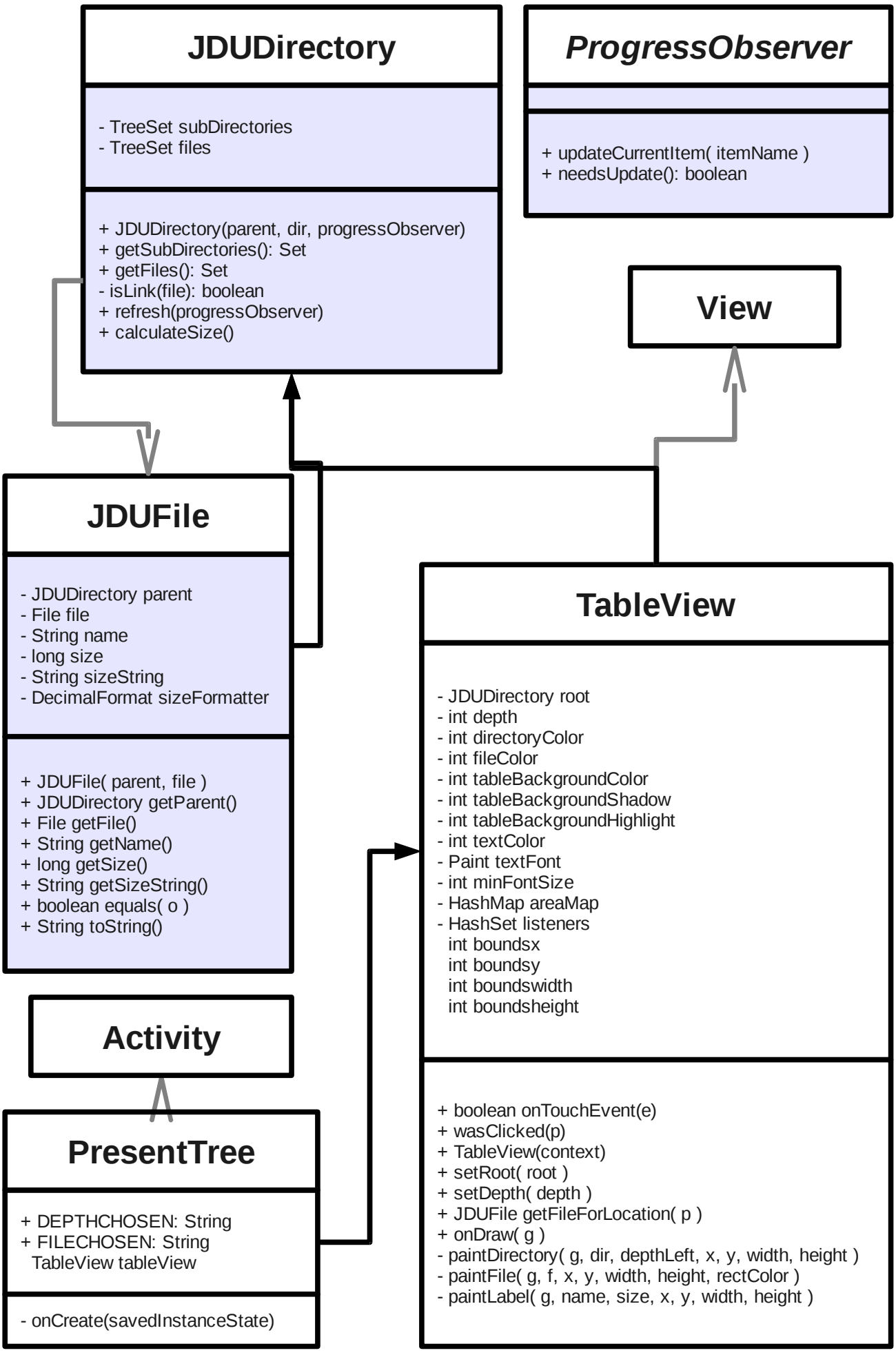
For at få det hele til at virke, skal der være nogle ”kedelige” klasser: `StartScreen`, `Help` og `About` er der ikke så mange ben i. De er alle underklasser til `Activity`. 1 af dem implementerer desuden `OnClickListener`, så de kan fange klik: hvilken aktivitet går man videre til fra start-skærmen, og hvilken dybde vælger man?

`DepthSetting` er lidt spændende, fordi den er underklasse til `PreferenceActivity` og placerer information om dybden.

`AndroidFileBrowser` er lånt andetsteds fra, og beskrives blot her.



Senere vil jeg tale om, at jeg har brug for en adapter, der kan vise en liste over mapper. Jeg havde dog specielle behov, og var derfor også nødt til at låne en adapter, jeg kunne tilpasse. Ud af det kom der `AfbOption` (der kan indeholde den information, den enkelte linje i listen senere skal vise) og `AfbAdapter`.


JDUDirectory
- TreeSet subDirectories
- TreeSet files
+ JDUDirectory(parent, dir, progressObserver)
+ getSubDirectories(): Set
+ getFiles(): Set
- isLink(file): boolean
+ refresh(progressObserver)
+ calculateSize()
ProgressObserver
+ updateCurrentItem(itemName)
+ needsUpdate(): boolean
View
JDUFile
- JDUDirectory parent
- File file
- String name
- long size
- String sizeString
- DecimalFormat sizeFormatter
+ JDUFile(parent, file)
+ JDUDirectory getParent()
+ File getFile()
+ String getName()
+ long getSize()
+ String getSizeString()
+ boolean equals(o)
+ String toString()
TableView
- JDUDirectory root
- int depth
- int directoryColor
- int fileColor
- int tableBackgroundColor
- int tableBackgroundShadow
- int tableBackgroundHighlight
- int textColor
- Paint textFont
- int minFontSize
- HashMap areaMap
- HashSet listeners
int boundsx
int boundsy
int boundswidth
int boundsheight
+ boolean onTouchEvent(e)
+ wasClicked(p)
+ TableView(context)
+ setRoot(root)
+ setDepth(depth)
+ JDUFile getFileForLocation(p)
+ onDraw(g)
- paintDirectory(g, dir, depthLeft, x, y, width, height)
- paintFile(g, f, x, y, width, height, rectColor)
- paintLabel(g, name, size, x, y, width, height)
Activity
PresentTree
+ DEPTHCHOSEN: String
+ FILECHOSEN: String
TableView tableView
- onCreate(savedInstanceState)

Nu kommer vi til de klasser, der faktisk laver alt arbejdet. Nogle klasser er overtaget stort set uændret: `JDUDirectory`, `JDUFile` og `ProgressObserver`, som allerede nævnt. (I praksis foretager `ProgressObserver` sig ikke noget længere, men det var lettere at beholde den, end at skulle skrive den ud af koden).

Når `JDUDirectory` er klar, indeholder den data om filsystemet i et træ. Herefter omsætter `TableView` disse data til et view. Arbejdet med at tegne træet på skærmen foretages af metoderne `paintDirectory()`, `paintFile()` og `paintLabel()`. `getFileForLocation()` finder ud af hvilken mappe, der er blevet tappet på, og sætter derefter den rigtige handling i gang.

`onTouchEvent()` opfanger klik, og giver dem videre til `wasClicked()` (koden kunne effektiviseres her, ved at lægge de to metoder sammen).

`PresentTree` modtager data om startmappe og dybde, sørger for at `JDUDirectory` bliver kaldt med disse parametre, og får siden `TableView` til at vise resultatet frem.

Programmering

DepthSetting

En underklasse til `PreferenceActivity`. 3 muligheder præsenteres via `settings.xml` og `arrays.xml`. Her bruges metoder knyttet til menuer (`addPreferencesFromResource()`, `getDefaultSharedPreferences()`), så vi kan huske hvilken dybde, der blev valgt.

I JDU valgte man dybde i en dropdown menu. Det vælger jeg ikke at gøre i ADU. Mest af alt fordi der ikke er behov for at kunne vælge helt op til dybde 8.

Splash

Denne første klasse er underklasse til `Activity`. Det mest interessante ved den er, at den har en `SensorEventListener`, der er sat op til at lytte efter, om telefonen vender og drejer. Hvis den gør det (og det gør den jo altid i en eller anden grad), vil denne information blive givet videre til `SplashView`.

`SplashView` ligner den nedenfor nævnte `TableView` meget, og er faktisk en kannibaliseret version af samme. Med et tap kommer man videre til den rigtige startskærm. Metoden `newCoords()` ændrer på positionen af den kasse, der bliver tegnet på skærmen, og tegner den igen. (Lille problem lige nu: hører kun y-koordinaten?)

StartScreen

Denne klasse er underklasse til `Activity` , og implementerer `OnClickListener` , sidstnævnte så brugeren kan tappe på knapperne og vælge hvad der nu skal ske.

Klassen indeholder en `onCreate()`, der via layout-filen `start.xml` viser 3 knapper frem og tilknytter en lytter til samme. `onClick()` opfanger klik, og gør noget passende afhængig af klikket.

About

Endnu en underklasse til `Activity`. En layout-fil, `about2.xml`, der fortæller noget om programmets oprindelse, vises. I manifestet er angivet, at denne fil vises som en dialog, dvs. henover

den bagvedliggende aktivitet, der gråes lidt ud, men ellers er synlig. Via et webview vises en html-side.

Help

Gør tingene på næsten fuldstændig samme måde som About, via `help.xml`, dog uden webview.

AndroidFileBrowser

Her er tale om en underklasse til `ListActivity` , så der kan blive vist en liste, i dette tilfælde over mapper.

`onCreate()` skulle gerne gemme hvilken mappe, man har fået bladet ned i, så det ikke gør noget, telefonen bliver vendt om. Dette virker i skrivende stund ikke endnu.

Der er metoder, der tillader at man kan vandre rundt i filsystemet. Desuden vil metoden `fill()` danne en liste over de mapper, man kan se netop nu. Den bliver sandsynligvis kaldt mere end én gang undervejs.

Den almindelige adapter egner sig rigtig godt til at få vist noget, der indeholder én variabel tekst pr. linje. Research viste, at hvis man skulle noget mere avanceret, skulle man tilsidesætte klassen `ArrayAdapter`, og lave sin egen.

Jeg faldt på nettet over en adapter, der kunne det jeg havde brug for (den viste 2 tekster og 1 billede), og som kunne tilrettes mine formål. Det blev til `AfbAdapter` (der altså er en underklasse til `ArrayAdapter`) og `AfbOption` (der indeholder de data, en enkelt linje har brug for).

Som jeg bruger `AfbAdapter`, lader jeg hver linje indeholde 2 stk. information, den ene ordet enten ”Analyze” eller ”Enter”, den anden stien til mappen. `afb_list_item.xml` viser en enkelt linje.

`onListItemClick()` er blevet væsentligt ændret. Oprindeligt skulle den blade frem til en fil, og så vise den. Jeg har brug for slet ikke at vise filer, blade frem til en mappe, og sende navnet på den videre. Desuden skal den kunne holde styr på, at hver mappe bliver vist 2 gange.

Mappenavnet sendes videre via `putExtra()`, der lader et `Intent` få ekstra information med.

PresentTree

Den sidste underklasse til `Activity`. Den gør ikke meget andet end at kalde `TableView`, og vise det resulterende View frem.

Ganske kort: `JDUDirectory` afsøger filsystemet og skaber en datamodel af samme. Her er der behov for at skelne mellem `JDUFile` og `JDUDirectory`. Et `JDUFile`-objekt har et `JDUDirectory` som parent (en fil ligger i en mappe), mens et `JDUDirectory` bl.a. også har undermapper og filer.

`ProgressObserver` er der stadig af historiske årsager, men bliver i praksis ikke brugt.

`TableView` er ændret ganske meget, da den grafiske del er helt anderledes på Android. Der er tilføjet en `onTouchEvent()`, for at fange tap (`TableView` havde en anden mekanisme til at opfange klik i vindues-miljøet). `getFileForLocation()` sammenligner koordinaterne med tappet med de tegnede kasser, for at finde den rigtige kasse og dermed den rigtige mappe at fokusere på fremover.

`onDraw()` erstatter også en tidligere metode. Det er desuden her der måles på, hvor stort mit canvas er, så jeg senere kan bruge de mål, når alle kasserne skal tegnes.

Så har der også været lidt roderi med, om rektangler måles på hvor bredde/høje de er, eller om der gives koordinater for nederste, højre hjørne.

Den del af koden, der tilpasser teksten til kasserne, springes helt over. Rigtig meget arbejde, der ikke beviser ret megen ny Android-viden.

Andet

Mit `AndroidManifest.xml` peger bl.a. på `jdulogo.png` som ikon for applikationen.

Udover hvad man normalt arbejder med på Android (`colors.xml`, `strings.xml`) fulgte der en `styles.xml` med med `afb_list_item.xml`.

Hovedkonklusion

En af de forhindringer, der for alvor gav mig bryderier var: hvordan håndterer man et filsystem, der har symbolske links, og derfor i princippet kan have mapper i uendelig dybde? Det lykkedes ikke at finde mig en løsning, men til gengæld fandt jeg mange tegn på, at andre har haft samme problem. Metoden `isLink()` var et af mine forsøg på at løse problemet.

Andre ting har nok primært været svære, fordi, nå ja, nogle gange ser man ikke lige det, der er for næsen af en. Således er min splash-skærm ikke blevet helt færdig, fordi jeg ikke mestrer det at bruge sensorer, og jeg har en menu-skærm, der ikke vil komme frem som en menu.

Kildetekster

assets/gettingStarted.html

```
<html>
<head>
  <title>ADU Getting Started</title>
</head>

<body style="background-color:#0089fa;">

<h1>The Absolute Basics:</h1>

<blockquote>
<font size=+1>
<p>
<b>1.</b> Click "Go!" to select a start directory.
</p>

<p>
<b>2.</b> Wait while ADU scans the directories and files.
</p>

<p>
<b>3.</b> Examine the resulting table, which shows how your disk space is being used
<ul>
  <li>you can click on subdirectories to examine them in more detail, and you
    can click on the left-most directory to expand your view one level</li>
  <li>If you are unhappy with the default depth of 3 (3 columns), choose
    menu to change it.</li>
</ul>
</p>

<p>
<b>4.</b> For more information,
visit  <a href="http://jasonpratt.org/software/jdu/">http://jasonpratt.org/software/jdu/</a></li>
</p>
</font>
</blockquote>

</body>
</html>
```

res/drawable/judlogo.png

res/layout/about.xml

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:padding="10dip">
   <TextView
      android:id="@+id/about_content"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="@string/about_text" />
</ScrollView>
```

res/layout/afb_list_item.xml

```
<?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="wrap_content"
  android:paddingLeft="8dip"
  android:paddingRight="8dip"
  android:paddingTop="5dip"
  android:paddingBottom="8dip"
  android:orientation="horizontal"
  >
```

```
  <TextView
    android:id="@+id/directoryaction"
    android:layout_width="100dip"
    android:layout_height="wrap_content"
    android:textAppearance="@style/genericListItemFirstTextView"
  />
  <TextView
    android:id="@+id/directoryname"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="@style/genericListItemSecondTextView" />
</LinearLayout>
```

res/layout/help2.xml

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:padding="10dip">
   <WebView
      android:id="@+id/web_view"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:layout_weight="1.0" />
</ScrollView>
```

res/layout/start.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:background="@color/background"
   android:layout_height="fill_parent"
   android:layout_width="fill_parent"
   android:padding="30dip"
   android:orientation="horizontal">
   <LinearLayout
      android:orientation="vertical"
      android:layout_height="wrap_content"
      android:layout_width="fill_parent"
      android:layout_gravity="center">
      <TextView
         android:text="@string/main_title"
         android:layout_height="wrap_content"
         android:layout_width="wrap_content"
         android:layout_gravity="center"
         android:layout_marginBottom="25dip"
         android:textSize="24.5sp" />
      <Button
         android:id="@+id/go_button"
         android:layout_width="fill_parent"
         android:layout_height="wrap_content"
         android:text="@string/go_label" />
      <Button
         android:id="@+id/about_button"
         android:layout_width="fill_parent"
         android:layout_height="wrap_content"
         android:text="@string/about_label" />
      <Button
         android:id="@+id/help_button"
         android:layout_width="fill_parent"
         android:layout_height="wrap_content"
         android:text="@string/help_label" />
      <Button
         android:id="@+id/depth_button"
         android:layout_width="fill_parent"
         android:layout_height="wrap_content"
         android:text="@string/depth_label" />
   </LinearLayout>
</LinearLayout>
```

res/values/arrays.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
 <string-array name="listArray">
     <item>2</item>
     <item>3</item>
     <item>4</item>
 </string-array>
 <string-array name="listValues">
     <item>2</item>
     <item>3</item>
     <item>4</item>
 </string-array>
</resources>
```

res/values/colors.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <color name="background">#ff0089fa</color>
   <color name="dirbackground">#ffb8d6ff</color>
   <color name="filebackground">#ffffe0c7</color>
   <color name="text">#ff000000</color>

  <color name="white">#fff</color>
  <color name="black">#000</color>
  <color name="red">#c6360a</color>
  <color name="green">#688f2b</color>
  <color name="orange">#f48905</color>
  <color name="dark_blue">#003366</color>
  <color name="grey">#888888</color>
</resources>
```

res/values/strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!--
 ! Excerpted from "Hello, Android! 3e",
 ! published by The Pragmatic Bookshelf.
 ! Copyrights apply to this code. It may not be used to create training material,
 ! courses, books, articles, and the like. Contact us if you are in doubt.
 ! We make no guarantees that this code is fit for any purpose.
 ! Visit http://www.pragmaticprogrammer.com/titles/eband3 for more book information.
-->
<resources>
   <string name="app_name">ADU</string>
   <string name="main_title">Android Disk Usage</string>
   <string name="continue_label">Continue</string>

   <string name="go_label">Go!</string>
   <string name="about_label">About</string>
   <string name="help_label">Help</string>

   <string name="depth_title">Choose depth</string>
   <string name="depth_label">Choose depth (tmp)</string>
   <string name="depth_text">The next step may take a few seconds</string>
   <string name="two_label">2</string>
   <string name="three_label">3</string>
   <string name="four_label">4</string>

   <string name="about_text">\
ADU: Android Disk Usage\nBased on JDU: Disk Usage,
version 1.1, http://jasonpratt.org/software/jdu/, (C) 2001,
Jason Pratt.\nAlso uses AndroidFileBrowser,
http://www.anddev.org/viewtopic.php?t=67 , and a special
ArrayAdapter,
http://trace.adityalesmana.com/2010/08/customize-android-listview-via-listadapter/ .
Ported by Lise Andreasen.
   </string>
   <string name="help_text">\
The Absolute Basics:\n1. Click "Go!" to select a start directory. Then choose a
depth.\n2. Wait while ADU scans the directories and files.\n3. Examine the
resulting table, which shows how your disk space is being used\n* you can
click on subdirectories to examine them in more detail, and you
    can click on the left-most directory to expand your view one level\n4. For
    more information, visit  http://jasonpratt.org/software/jdu/
   </string>
    <string name="up_one_level">..</string>
    <string name="current_dir">.</string>

    <string name="analyze_label">Analyze</string>
    <string name="enter_label">Enter</string>
   <string name="test_title">Hello world</string>
</resources>
```

res/values/styles.xml

```
<?xml version="1.0" encoding="UTF-8"?>
  <!-- Styling file -->
<resources>
  <style name="listSeparatorTextViewStyle" parent="@android:attr/listSeparatorTextViewStyle">
    <item name="android:layout_height">wrap_content</item>
    <item name="android:layout_width">fill_parent</item>
    <item name="android:textSize">15dip</item>
    <item name="android:paddingTop">2dip</item>
    <item name="android:paddingBottom">3dip</item>
    <item name="android:paddingLeft">5dip</item>
    <item name="android:paddingRight">10dip</item>
    <item name="android:textAppearance">@android:style/TextAppearance.Small</item>
    <item name="android:shadowColor">#111111</item>
    <item name="android:shadowRadius">1</item>
    <item name="android:shadowDy">1</item>
    <item name="android:textStyle">bold</item>
    <item name="android:textColor">@android:color/white</item>
  </style>
```

```
  <style name="listViewStyle" parent="@android:attr/listViewStyle">
    <item name="android:textColor">@android:color/white</item>
  </style>
    <style name="genericListItemFirstTextView">
        <item name="android:textSize">15dip</item>
        <item name="android:textStyle">bold</item>
        <item name="android:textColor">@color/white</item>
    </style>
    <style name="genericListItemSecondTextView">
        <item name="android:textSize">13dip</item>
        <item name="android:textColor">@color/white</item>
    </style>
</resources>
```

res/sml/settings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen
  xmlns:android="http://schemas.android.com/apk/res/android">
  <ListPreference
     android:key="listPref"
     android:title="ADU depth"
     android:summary="Choose the depth for the ADU presenter"
     android:entries="@array/listArray"
     android:entryValues="@array/listValues" />
</PreferenceScreen>
```

src/org/ViewCreateTestActivity/AfbAdapter.java

```
package org.ViewCreateTestActivity;

import android.content.Context;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.ImageView;
import android.widget.TextView;
import java.util.ArrayList;
import org.me.aduapplication.R;

/*
 * Borrowed from
 * http://trace.adityalesmana.com/2010/08/customize-android-listview-via-listadapter/
 */

public class AfbAdapter extends ArrayAdapter<AfbOption> {
    private ArrayList<AfbOption> tests;
    private LayoutInflater mInflater;
    private int layoutResource;
    public AfbAdapter(Context context, int textViewResourceId,
            ArrayList<org.ViewCreateTestActivity.AfbOption> mOptions) {
        super(context, textViewResourceId, mOptions);
        this.tests = mOptions;
        this.mInflater = LayoutInflater.from(context);
        this.layoutResource = textViewResourceId;
    }
    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        final ViewHolder holder;
        View v = convertView;
        if (v == null) {
            v = mInflater.inflate(layoutResource, null);
            holder = new ViewHolder();
            holder.firstLine = (TextView) v.findViewById(R.id.directoryaction);
            holder.secondLine = (TextView) v.findViewById(R.id.directoryname);
            v.setTag(holder);
        }else {
            // Get the ViewHolder back to get fast access to the TextView
            // and the ImageView.
            holder = (ViewHolder) v.getTag();
        }
        AfbOption c = tests.get(position);
        if (c != null) {
            //loading first line
            holder.firstLine.setText(c.getDescription());
            holder.firstLine.setVisibility(View.VISIBLE);
            //loading second line
            holder.secondLine.setText(c.getName());
            holder.secondLine.setVisibility(View.VISIBLE);
        }
        //todo create stringformatter
        return v;
    }
    private static class ViewHolder {
        TextView firstLine;
        TextView secondLine;
    }
}
```

src/org/ViewCreateTestActivity/AfbOption.java

```
package org.ViewCreateTestActivity;

/*
 * Borrowed from
 * http://trace.adityalesmana.com/2010/08/customize-android-listview-via-listadapter/
 */

public class AfbOption{
    // 2 texts for a later ListActivity item
    private String name;
    private String description;
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getDescription() {
        return description;
    }
    public void setDescription(String description) {
        this.description = description;
    }
}
```

src/org/anddev/AndroidFileBrowser.java

```
/*
 * borrowed from
 * http://www.anddev.org/viewtopic.php?t=67
 * Building an Android FileBrowser (list-based) ! (novice tutorial)
 */
package org.anddev;
import java.io.File;
import java.io.IOException;
import java.util.ArrayList;

import android.app.ListActivity;
import android.content.Intent;
import android.os.Bundle;
// import android.util.Log;
import android.util.Log;
import android.view.View;
import android.widget.ListView;
import java.util.Collections;
import java.util.logging.Level;
import java.util.logging.Logger;
import org.ViewCreateTestActivity.AfbAdapter;
import org.ViewCreateTestActivity.AfbOption;
import org.me.aduapplication.PresentTree;
import org.me.aduapplication.R;

public class AndroidFileBrowser extends ListActivity {

     private enum DISPLAYMODE{ ABSOLUTE, RELATIVE; }

     private final DISPLAYMODE displayMode = DISPLAYMODE.ABSOLUTE;
     private ArrayList<String> directoryEntries = new ArrayList<String>();
     private File currentDirectory;

     /** Called when the activity is first created. */
     @Override
     public void onCreate(Bundle icicle) {
          super.onCreate(icicle);

          if (icicle == null) {
            currentDirectory = new File("/");
          } else {
            currentDirectory = (File) icicle.getSerializable("currentDirectory");
          }

          // setContentView() gets called within the next line,
          // so we do not need it here.

          browseTo(currentDirectory);
     }

     @Override
     protected void onRestoreInstanceState (Bundle icicle) {
         // her genskabes indhold for alle views med id
       super.onRestoreInstanceState(icicle);
     }

     @Override
     protected void onSaveInstanceState(Bundle icicle) {
        super.onSaveInstanceState(icicle); // gem indhold for alle views med id
        icicle.putSerializable("currentDirectory", currentDirectory);
     }
```

```
/**
 * This function browses up one level
 * according to the field: currentDirectory
 */
private void upOneLevel(){
     if(this.currentDirectory.getParent() != null)
          this.browseTo(this.currentDirectory.getParentFile());
}

private void browseTo(final File aDirectory){
   AndroidFileBrowser.this.currentDirectory = aDirectory;
   try {
     fill(aDirectory.listFiles());
   } catch (IOException ex) {
     Logger.getLogger(AndroidFileBrowser.class.getName()).log(Level.SEVERE, null, ex);
   }

}

// create list of ittems, display them
private void fill(File[] files) throws IOException {
     String tmpString;
     this.directoryEntries.clear();
     ArrayList<AfbOption> m_options = null;

     // Add the "." and the ".." == 'Up one level'
     try {
          Thread.sleep(10);
     } catch (InterruptedException e1) {
     }
     this.directoryEntries.add(".");

     if(this.currentDirectory.getParent() != null) {
          this.directoryEntries.add("..");
     }

     switch(this.displayMode){
          case ABSOLUTE:
               for (File file : files){
                 if (file.isDirectory()) {
                    this.directoryEntries.add(file.getPath());
                 }
               }
               break;
          case RELATIVE: // On relative Mode, we have to add the current-path to the beginning
               int currentPathStringLenght = this.currentDirectory.getAbsolutePath().length();
               for (File file : files){
                 if (file.isDirectory()) {
                    tmpString = file.getAbsolutePath().substring(currentPathStringLenght);
                    this.directoryEntries.add(tmpString);
                 }
               }
               break;
     }

     Collections.sort(directoryEntries);

     AfbAdapter directoryList;
     m_options = new ArrayList<AfbOption>();
     AfbOption act_dir;

     int len = directoryEntries.size();

     for (int i = 0; i < len; ++i) {
       act_dir = new AfbOption();
       act_dir.setName(directoryEntries.get(i));
       act_dir.setDescription("Analyze");
       m_options.add(act_dir);

       act_dir = new AfbOption();
       act_dir.setName(directoryEntries.get(i));
       act_dir.setDescription("Enter");
       m_options.add(act_dir);
     }

     // TODO deferred: list item same colors as rest of application
     directoryList = new AfbAdapter(this, R.layout.afb_list_item, m_options);
     this.setListAdapter(directoryList);
}

// adapted to ADU
@Override
protected void onListItemClick(ListView l, View v, int position, long id) {
     int selectionRowID = position / 2;
     int selectedAction = position % 2; // 0: analyze, 1: enter
     String selectedFileString = this.directoryEntries.get(selectionRowID);

     File clickedFile = null;
     if (! selectedFileString.equals(".") && !selectedFileString.equals("..")){
        switch(this.displayMode){
             case RELATIVE:
                  clickedFile = new File(this.currentDirectory.getAbsolutePath()
                                                 + this.directoryEntries.get(selectionRowID));
                  break;
             case ABSOLUTE:
                  clickedFile = new File(this.directoryEntries.get(selectionRowID));
                  break;
        }
     }
```

```
          if (selectedAction == 0) { // analyze
            Intent myIntent;
            myIntent = new Intent(AndroidFileBrowser.this, PresentTree.class);

            String myString = "";

            if (selectedFileString.equals(".")) { // .
               myString = selectedFileString;
            } else if(selectedFileString.equals("..")){ // ..
              if(this.currentDirectory.getParent() != null) {
                myString = this.currentDirectory.getParentFile().getAbsolutePath();
              } else {
                myString = this.currentDirectory.getAbsolutePath();
              }
            } else { // not . not ..
               if(clickedFile != null) {
                    myString = clickedFile.getAbsolutePath();
               }
            }
            myIntent.putExtra(PresentTree.FILECHOSEN, myString);

            startActivity(myIntent);
          } else { // enter
            if (selectedFileString.equals(".")) { // .
               // Refresh
               this.browseTo(this.currentDirectory);
            } else if(selectedFileString.equals("..")){ // ..
               this.upOneLevel();
            } else { // not . not ..
               this.browseTo(clickedFile);
            }
          }
      }
}
```

src/org/jasonpratt/jdu/JDUDirectory

```
/*
 * JDU: Disk Usage
 *
 * Copyright (c) 2001-2005 Jason Pratt
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
 * copy of this software and associated documentation files (the "Software"),
 * to deal in the Software without restriction, including without limitation
 * the rights to use, copy, modify, merge, publish, distribute, sublicense,
 * and/or sell copies of the Software, and to permit persons to whom the
 * Software is furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
 * THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
 * FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
 * DEALINGS IN THE SOFTWARE.
 */

package org.jasonpratt.jdu;

import android.util.Log;
import java.io.File;
import java.io.IOException;

/**
 * @author Jason Pratt
 */
public class JDUDirectory extends JDUFile {
  protected java.util.Comparator fileComparator = new java.util.Comparator() {
    public int compare( Object o1, Object o2 ) {
      if( (o1 instanceof JDUFile) && (o2 instanceof JDUFile) ) {
        long size1 = ((JDUFile)o1).getSize();
        long size2 = ((JDUFile)o2).getSize();
        if( size1 < size2 ) {
          return 1;
        } else if( size1 > size2 ) {
          return -1;
        } else {
          return 0;
        }
      }

      return 0; // if it's not a File, don't bother
    }
  };
  protected java.util.TreeSet subDirectories = new java.util.TreeSet( fileComparator );
  protected java.util.TreeSet files = new java.util.TreeSet( fileComparator );

  public JDUDirectory( JDUDirectory parent, java.io.File dir, ProgressObserver progressObserver ) {
    super( parent, dir );
    refresh( progressObserver );
  }
```

```
  public java.util.Set getSubDirectories() {
    return subDirectories;
  }

  public java.util.Set getFiles() {
    return files;
  }

  // TODO problem if start directory is / /proc /sdcard /sys

  /*
   * borrowed from http://www.idiom.com/~zilla/Xfiles/javasymlinks.html
   */
  private static boolean isLink(File file)
  {
    try {
      if (!file.exists())
        return true;
      else
      {
        String cnnpath = file.getCanonicalPath();
        String abspath = file.getAbsolutePath();
        return !abspath.equals(cnnpath);
      }
    }
    catch(IOException ex) {
      System.err.println(ex);
      return true;
    }
  } //isLink


  synchronized public void refresh( ProgressObserver progressObserver ) {
    if( (progressObserver != null) && progressObserver.needsUpdate() ) {
      progressObserver.updateCurrentItem( getFile().getAbsolutePath() );
    }

    java.io.File dir = getFile();
    subDirectories.clear();
    files.clear();

    java.io.File[] children = dir.listFiles();
    if( children != null ) {
      for( int i = 0; i < children.length; i++ ) {
        // if (!isLink(children[i])) {
          if( children[i].isDirectory() ) {
            subDirectories.add( new JDUDirectory( this, children[i], progressObserver ) );
          } else { // not a directory
            if( children[i].isFile() ) {
              files.add( new JDUFile( this, children[i] ) );
            } else {
              System.err.println( "Not a directory or a file: " + children[i].getAbsolutePath() );
            } // isFile
          }
        // } // !isLink
      }
    }

    if( (progressObserver != null) && progressObserver.needsUpdate() ) {
      progressObserver.updateCurrentItem( getFile().getAbsolutePath() );
    }
    calculateSize();
  }

  public void calculateSize() {
    size = 0;
    for( java.util.Iterator iter = subDirectories.iterator(); iter.hasNext(); ) {
      JDUDirectory d = (JDUDirectory)iter.next();
      size += d.getSize();
    }
    for( java.util.Iterator iter = files.iterator(); iter.hasNext(); ) {
      JDUFile f = (JDUFile)iter.next();
      size += f.getSize();
    }
  }
}
```

src/org/jasonpratt/jdu/JDUFile

```
/*
 * JDU: Disk Usage
 *
 * Copyright (c) 2001-2005 Jason Pratt
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
 * copy of this software and associated documentation files (the "Software"),
 * to deal in the Software without restriction, including without limitation
 * the rights to use, copy, modify, merge, publish, distribute, sublicense,
 * and/or sell copies of the Software, and to permit persons to whom the
 * Software is furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
```

```
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
 * THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
 * FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
 * DEALINGS IN THE SOFTWARE.
 */

package org.jasonpratt.jdu;

/**
 * @author Jason Pratt
 */
public class JDUFile {
  protected JDUDirectory parent;
  protected java.io.File file;
  protected String name;
  protected long size = -1;
  protected String sizeString;
  protected java.text.DecimalFormat sizeFormatter = new java.text.DecimalFormat( "#0.00" );

  public JDUFile( JDUDirectory parent, java.io.File file ) {
    this.parent = parent;
    this.file = file;
    if( file != null ) {
      this.name = file.getName();
      this.size = file.length();
    }
  }

  public JDUDirectory getParent() {
    return parent;
  }

  public java.io.File getFile() {
    return file;
  }

  public String getName() {
    if( name == null ) { // delayed lookup
      name = file.getName();
    }
    return name;
  }

  public long getSize() {
    if( size == -1 ) { // delayed lookup
      size = file.length();
    }
    return size;
  }

  public String getSizeString() {
    if( sizeString == null ) { // create on demand
      getSize(); // ensure size is up-to-date
      if( size < 1024 ) {
        sizeString = sizeFormatter.format( size ) + " bytes";
      } else if( size < 1024L*1024L ) {
        sizeString = sizeFormatter.format( ((double)size)/((double)1024) ) + " KB";
      } else if( size < 1024L*1024L*1024L ) {
        sizeString = sizeFormatter.format( ((double)size)/((double)1024L*1024L) ) + " MB";
      } else if( size < 1024L*1024L*1024L*1024L ) {
        sizeString = sizeFormatter.format( ((double)size)/((double)1024L*1024L*1024L) ) + " GB";
      } else {
        sizeString = sizeFormatter.format( ((double)size)/((double)1024L*1024L*1024L*1024L) ) + " TB";
      }
    }

    return sizeString;
  }

  public boolean equals( Object o ) {
    if( o instanceof JDUFile ) {
      return ((JDUFile)o).getFile().equals( file );
    } else {
      return file.equals( o );
    }
  }

  public String toString() {
    if( file == null ) {
      return "null";
    } else {
      return file.getAbsolutePath();
    }
  }
}
```

src/org/jasonpratt/jdu/ProgressObserver.java

```
/*
 * JDU: Disk Usage
 *
 * Copyright (c) 2001-2005 Jason Pratt
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
 * copy of this software and associated documentation files (the "Software"),
```

```
 * to deal in the Software without restriction, including without limitation
 * the rights to use, copy, modify, merge, publish, distribute, sublicense,
 * and/or sell copies of the Software, and to permit persons to whom the
 * Software is furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
 * THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
 * FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
 * DEALINGS IN THE SOFTWARE.
 */

package org.jasonpratt.jdu;

/**
 * @author Jason Pratt
 */
public interface ProgressObserver {
  public void updateCurrentItem( String itemName );
  public boolean needsUpdate();
}
```

src/org/jasonpratt/jdu/TableView.java

```
/*
 * JDU: Disk Usage
 *
 * Copyright (c) 2001-2005 Jason Pratt
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
 * copy of this software and associated documentation files (the "Software"),
 * to deal in the Software without restriction, including without limitation
 * the rights to use, copy, modify, merge, publish, distribute, sublicense,
 * and/or sell copies of the Software, and to permit persons to whom the
 * Software is furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
 * THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
 * FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
 * DEALINGS IN THE SOFTWARE.
 */

package org.jasonpratt.jdu;

import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.Point;
import android.graphics.Rect;
import android.graphics.Typeface;
// import android.util.Log;
import android.view.MotionEvent;
import android.view.View;

/**
 * @author Jason Pratt
 */
public class TableView extends View {
  // private static final long serialVersionUID = 873755198917107930L;
  protected JDUDirectory root;
  protected int depth = 3;
  protected int directoryColor = Color.argb(255, 200, 230, 255 );
  protected int fileColor = Color.argb(255, 255, 240, 215 );
  protected int tableBackgroundColor = Color.argb(255, 96, 96, 96 );
  protected int tableBackgroundShadow = Color.argb(255, 32, 32, 32 );
  protected int tableBackgroundHighlight = Color.argb(255, 196, 196, 196 );
  protected int textColor = Color.argb(255, 0, 0, 0);
  protected Paint textFont = new Paint(); // "SansSerif", Font.PLAIN, 14 );
  protected int minFontSize = 4;
  protected java.util.HashMap areaMap = new java.util.HashMap();
  protected java.util.HashSet listeners = new java.util.HashSet();
  protected int boundsx, boundsy, boundswidth, boundsheight;

    @Override
    public boolean onTouchEvent(MotionEvent e)
    {
      if (e.getAction() == MotionEvent.ACTION_DOWN) {
        Point p = new Point(1, 2);
        p.x = (int) e.getX();
        p.y = (int) e.getY();
        this.wasClicked(p);
      }
      return true;
    }
```

```
  public void wasClicked(Point p)  {
      JDUFile f = getFileForLocation( p );
      if( f instanceof JDUDirectory ) {
          JDUDirectory d = (JDUDirectory)f;
          if( TableView.this.root == d ) {
              if( d.getParent() != null ) {
                  TableView.this.setRoot( d.getParent() );
                  invalidate();
              }
          } else {
              TableView.this.setRoot( d );
              invalidate();
          }
      }
  }

//////////////////
// Constructor
//////////////////

public TableView(Context context) {
    super(context);
    setBackgroundColor(tableBackgroundColor);
}

//////////////
// utility
//////////////

public void setRoot( JDUDirectory root ) {
  JDUDirectory oldRoot = this.root;
  this.root = root;
  //fireRootChanged( oldRoot, root );
  invalidate();
}

public void setDepth( int depth ) {
  this.depth = depth;
  invalidate();
}

public JDUFile getFileForLocation( Point p ) {
  // relate click to directory
  if( (p.x >= boundsx) && (p.x < boundswidth+boundsx) && (p.y >= boundsy) && (p.y < boundsheight+boundsy) ) {
    for( java.util.Iterator iter = areaMap.keySet().iterator(); iter.hasNext(); ) { // could be done more efficiently
      Rect bounds = (Rect)iter.next();
      if( bounds.contains( p.x + boundsx, p.y + boundsy) ) {
        Object o = areaMap.get( bounds );
        if( o instanceof JDUFile ) {
          return (JDUFile)o;
        } else {
          return null;
        }
      }
    }
  }

  return null;
}

////////////////
// Painting
////////////////

@Override
public void onDraw( Canvas g ) {
  super.onDraw(g);
  if( (root != null) && (depth > 0) ) {
    areaMap.clear();

    boundsx = this.getLeft();
    boundsy = this.getTop();
    boundswidth = this.getRight() - boundsx;
    boundsheight = this.getBottom() - boundsy;

    paintDirectory( g, root, depth, boundsx, boundsy, boundswidth, boundsheight );
    //paintDirectory( g, root, depth, bounds.x, bounds.y, bounds.width, bounds.height );
  }
}

protected void paintDirectory( Canvas g, JDUDirectory dir, int depthLeft, int x, int y, int width, int height ) {
  if( (width > 1) && (height > 1) ) {
    // TODO if painting dir with no subdirs ...
    int columnWidth = width/depthLeft;
    Paint rectColor = new Paint();
    rectColor.setColor(directoryColor);
    paintFile( g, dir, x, y, columnWidth, height, rectColor );
    areaMap.put( new Rect( x, y, x + columnWidth, y + height ), dir );

    if( depthLeft > 1 ) {
      int subX = x + columnWidth;
      int subY = y;
      int subWidth = width - columnWidth;
```

```
        for( java.util.Iterator iter = dir.getSubDirectories().iterator(); iter.hasNext(); ) {
          JDUDirectory subDir = (JDUDirectory)iter.next();
          long subHeight = ((long)height*subDir.getSize())/dir.getSize();
          if( subHeight < 2 ) {
            break;
          }
          paintDirectory( g, subDir, depthLeft - 1, subX, subY, subWidth, (int)subHeight );
          subY += subHeight;
        }

        for( java.util.Iterator iter = dir.getFiles().iterator(); iter.hasNext(); ) {
          JDUFile subFile = (JDUFile)iter.next();
          long subHeight = ((long)height*subFile.getSize())/dir.getSize();
          if( subHeight < 2 ) {
            break;
          }
          rectColor.setColor(fileColor);
          paintFile( g, subFile, subX, subY, subWidth/(depthLeft - 1), (int)subHeight, rectColor );
          areaMap.put( new Rect( subX, subY, x + subWidth/(depthLeft - 1), y + (int)subHeight ), subFile );
          subY += subHeight;
        }
      }
    }
  }

  protected void paintFile( Canvas g, JDUFile f, int x, int y, int width, int height, Paint rectColor ) {
    if( (width > 1) && (height > 1) ) {
      Paint edgeColor = new Paint();
      edgeColor.setColor(textColor); // text and edge should be same color
      g.drawRect(x, y, x+width, y+height, edgeColor);
      g.drawRect(x, y, x+width-1, y+height-1, rectColor);
      if( height >= 14 ) {
        String nameString = f.getName();
        if( nameString.trim().equals( "" ) ) {
          nameString = f.getFile().getAbsolutePath();
        }
        String sizeString = f.getSizeString();
        paintLabel( g, nameString, sizeString, x + 3, y + 15, width - 40, height - 2 );
      }
    }
  }

  protected void paintLabel( Canvas g, String name, String size, int x, int y, int width, int height ) {

      Paint letterColor = new Paint();
      letterColor.setColor(textColor);
      letterColor.setTypeface(Typeface.SANS_SERIF);
      letterColor.setTextSize(10);

      /*
       * TODO deferred: fit text
       * Used to be stuff here that fitted the text to the window
       */

      g.drawText(name + " " + size, x, y, letterColor);
  }

}
```

src/org/me/aduapplication/About.java

```
package org.me.aduapplication;

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

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

src/org/me/aduapplication/DepthSetting.xml

```
package org.me.aduapplication;

import android.content.Context;
import android.os.Bundle;
import android.preference.PreferenceActivity;
import android.preference.PreferenceManager;

// TODO why doesn't this activity appear, when I press the menu button?
public class DepthSetting extends PreferenceActivity {

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    addPreferencesFromResource(R.xml.settings);
  }
```

```
  /** Get the current value of the depth option */
  public static int getDepth(Context context) {
    return Integer.parseInt(PreferenceManager.getDefaultSharedPreferences(context).getString("listPref", "3"));
  }

}
```

src/org/me/aduapplication/Help.java

```
package org.me.aduapplication;

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

public class Help extends Activity {
   private WebView webView;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.help2);
      webView = (WebView) findViewById(R.id.web_view);
      webView.loadUrl("file:///android_asset/gettingStarted.html");   }
}
```

src/org/me/aduapplication/PresentTree.java

```
package org.me.aduapplication;

import android.app.Activity;
import android.os.Bundle;
// import android.util.Log;
import java.io.File;
import org.jasonpratt.jdu.JDUDirectory;
import org.jasonpratt.jdu.TableView;

public class PresentTree extends Activity {

  public static final String FILECHOSEN = "org.me.ADUapplication.file" ;
  public static final String DEPTHCHOSEN = "org.me.ADUapplication.depth" ;

  TableView tableView;

  /** Called when the activity is first created. */
  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    //create model of file system
    String fileChosen = getIntent().getStringExtra(FILECHOSEN);
    JDUDirectory myJDU = new JDUDirectory( null, new File(fileChosen), null );

    // create picture of model
    tableView = new TableView(this);

    int depthChosen = 3;
    depthChosen = DepthSetting.getDepth(this);
    // depthChosen = getIntent().getIntExtra(DEPTHCHOSEN, 2);
    tableView.setDepth(depthChosen);

    tableView.setRoot( myJDU );

    setContentView(tableView);
  }


}
```

src/org/me/aduapplication/Splash.java

```
package org.me.aduapplication;

import android.app.Activity;
import android.os.Bundle;
// import android.view.View;
import android.util.Log;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;

public class Splash extends Activity {

  SplashView sView;
  SensorManager sensorManager;
  Sensorlytter sensorlytter = new Sensorlytter();
```

```
  /** Called when the activity is first created. */
  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    sView = new SplashView(this);

    setContentView(sView);
  }

  @Override
  protected void onResume() {
      super.onResume();
      sensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
      sView = new SplashView(this);

      setContentView(sView);

      for (Sensor sensor : sensorManager.getSensorList(Sensor.TYPE_ALL)) {
        sensorManager.registerListener(sensorlytter, sensor, SensorManager.SENSOR_DELAY_NORMAL);
      }

  }

  @Override
  protected void onPause() {
      super.onPause();
      sensorManager.unregisterListener(sensorlytter);
  }

  class Sensorlytter implements SensorEventListener
  {
     public void onSensorChanged(SensorEvent event) {
      int sensortype = event.sensor.getType();

      if (sensortype == Sensor.TYPE_ORIENTATION)  {
        sView.newCoords(event.values[1],event.values[2]);
      }
    }

    public void onAccuracyChanged(Sensor sensor, int præcision) {
      // ignorér - men vi er nødt til at have metoden for at implementere interfaces
    }
  }; // Sensorlytter slut

}
```

src/org/me/aduapplication/SplashView.java

```
package org.me.aduapplication;

import android.content.Context;
import android.content.Intent;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
// import android.graphics.Point;
import android.graphics.Rect;
import android.graphics.Typeface;
import android.util.Log;
import android.view.MotionEvent;
import android.view.View;

public class SplashView extends View {
  protected int directoryColor = Color.argb(255, 200, 230, 255 );
  protected int tableBackgroundColor = Color.argb(255, 96, 96, 96 );
  protected int textColor = Color.argb(255, 0, 0, 0);
  protected int headingColor = Color.argb(255, 255, 0, 0);
  protected Paint textFont = new Paint(); // "SansSerif", Font.PLAIN, 14 );
  int boundsx, boundsy, boundswidth, boundsheight;
  int rectx, recty, rectwidth, rectheight;

  @Override
  public boolean onTouchEvent(MotionEvent e)
  {
    if (e.getAction() == MotionEvent.ACTION_DOWN) {
      Intent i = new Intent(getContext(), StartScreen.class);
      getContext().startActivity(i);
    }
    return true;
  }

  ///////////////////
  // Constructor
  ///////////////////

  public SplashView(Context context) {
      super(context);
      setBackgroundColor(tableBackgroundColor);
      rectx = -1;
  }

  ////////////////
  // Moving
  ////////////////
```

```
  public void newCoords(double pitch, double roll) {
    // TODO only up/down, not left/right?
    if (pitch > 0) {
      recty = recty - 10;
    } else {
      recty = recty + 10;
    }
    if (roll > 0) {
      rectx = rectx - 10;
    } else {
      rectx = rectx + 10;
    }
    if (rectx < boundsx) {
      rectx = boundsx;
    }
    if (recty < boundsy) {
      recty = boundsy;
    }
    if (boundswidth - rectwidth < rectx) {
      rectx = boundswidth - rectwidth;
    }
    if (boundsheight - rectheight < recty) {
      recty = boundsheight - rectheight;
    }
    invalidate();
  }

  ////////////////
  // Painting
  ////////////////

  @Override
  public void onDraw( Canvas g ) {
    super.onDraw(g);

    boundsx = this.getLeft();
    boundsy = this.getTop();
    boundswidth = this.getRight() - boundsx;
    boundsheight = this.getBottom() - boundsy;

    Paint letterColor = new Paint();
    letterColor.setColor(headingColor);
    letterColor.setTypeface(Typeface.SANS_SERIF);
    letterColor.setTextSize(20);
    g.drawText("Tap to move on", 100, 100, letterColor);

    if (rectx == -1) {
      // middle of the screen
      rectx = (int)(1.5 * boundswidth/4);
      recty = (int)(1.5 * boundsheight/4);
      rectwidth = (int)(boundswidth/4);
      rectheight = (int)(boundsheight/4);
    }

    paintDirectory( g, rectx, recty, rectwidth, rectheight );
  }

  protected void paintDirectory( Canvas g, int x, int y, int width, int height ) {
    Paint rectColor = new Paint();
    rectColor.setColor(directoryColor);
    paintFile( g, x, y, width, height, rectColor );
  }

  protected void paintFile( Canvas g, int x, int y, int width, int height, Paint rectColor ) {
    if( (width > 1) && (height > 1) ) {
      Paint edgeColor = new Paint();
      edgeColor.setColor(textColor); // text and edge should be same color
      g.drawRect(x, y, x+width, y+height, edgeColor);
      g.drawRect(x, y, x+width-1, y+height-1, rectColor);
      if( height >= 14 ) {
        String nameString = "/system";
        String sizeString = "100 KB";
        paintLabel( g, nameString, sizeString, x + 3, y + 15, width - 40, height - 2 );
      }
    }
  }

  protected void paintLabel( Canvas g, String name, String size, int x, int y, int width, int height ) {

      Paint letterColor = new Paint();
      letterColor.setColor(textColor);
      letterColor.setTypeface(Typeface.SANS_SERIF);
      letterColor.setTextSize(10);

      g.drawText(name + " " + size, x, y, letterColor);
  }

}
```

src/org/me/aduapplication/StartScreen.java

```
package org.me.aduapplication;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
```

```
import android.view.View;
import android.view.View.OnClickListener;
import org.anddev.AndroidFileBrowser;
// import android.util.Log;

public class StartScreen extends Activity implements OnClickListener {

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

      View aboutButton = findViewById(R.id.about_button);
      aboutButton.setOnClickListener(this);
      View helpButton = findViewById(R.id.help_button);
      helpButton.setOnClickListener(this);
      View goButton = findViewById(R.id.go_button);
      goButton.setOnClickListener(this);
      View depthButton = findViewById(R.id.depth_button);
      goButton.setOnClickListener(this);
  };

  public void onClick(View v) {
    Intent i;
    switch (v.getId()) {
    case R.id.about_button:
      i = new Intent(this, About.class);
      startActivity(i);
      break;
    case R.id.help_button:
      i = new Intent(this, Help.class);
      startActivity(i);
      break;
    case R.id.go_button:
      i = new Intent(this, AndroidFileBrowser.class);
      startActivity(i);
      break;
    // TODO this doesn't work either
    case R.id.depth_button:
      i = new Intent(this, DepthSetting.class);
      startActivity(i);
      break;
    }
  }
}
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
     package="org.me.aduapplication">
<uses-permission android:name="android.permission.INTERNET" />
    <application
      android:icon="@drawable/jdulogo"
      android:label="@string/app_name"
    >
         <activity android:name="org.me.aduapplication.DepthSetting" android:label="ADU settings" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>

         <activity android:name="org.me.aduapplication.Splash" android:label="ADU">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>

         <activity android:name="org.me.aduapplication.StartScreen" android:label="ADU start" />
         <activity android:name="org.anddev.AndroidFileBrowser" android:label="ADU file" />
         <activity android:name="org.me.aduapplication.PresentTree" android:label="ADU presenter" />
         <activity android:name="org.me.aduapplication.About" android:label="ADU about"
                   android:theme="@android:style/Theme.Dialog" />
         <activity android:name="org.me.aduapplication.Help" android:label="ADU help" android:theme="@android:style/Theme.Dialog" />

    </application>
</manifest>
```