1 Der Einstieg in Java für Android

Diese Ergänzung zum Buch „Programmieren in Java“ will Ihnen dabei helfen, erste Programme für Smartphones der Android–Plattform von Google zu erstellen und diese Programme ablaufen zu lassen. Diese Ergänzungen folgen dem Aufbau des o. a. Buches: in einzelnen Kapiteln zeigen wir, wie Sie Apps für Android entwickeln können.

Hinweise

Diese Ergänzung zum Buch „Programmieren in Java“ erläutert nur die Besonderheiten zum Einstieg in Android. Sie benötigen das o. a. Buch, welches hier kurz als „das Buch“ bezeichnet wird. Smartphones für die Android–Plattform sind kurz als Smartphones bezeichnet.

Smartphones sind batteriebetriebene Geräte. Der Stromverbrauch ist eine kritische Größe, die man nie aus dem Auge verlieren darf. Außerdem sind die Größen des RAM sowie des Hintergrundspeichers sowie die Rechenleistung im Vergleich zu PCs eng beschränkt. Diese Restriktionen erfordern in Teilen eine gegenüber dem PC erheblich aufwändigere Programmierung. Meine Empfehlung: starten Sie bei der Java-Programmierung mit dem Buch. Wenn Sie eine gewisse Sicherheit gewonnen haben, kann es viel Spaß machen, Apps für Android zu entwickeln. Dann können Sie diese Ergänzungen für Ihren persönlichen Start in die Welt der Androiden benützen.

Sie finden alle Programme zu dieser Ergänzung in der Form von Eclipse-Projekten unter der Homepage des Autors: http://www.hs-regensburg.de/~jof39108.

1.1 Wie entwickelt man Apps für Android?

Eine App für Android ist eine Applikation für das System Android. Eine solche App läuft im Prinzip auf einem Smartphone für Android. Wir entwickeln die App aber nicht auf dem Smartphone, sondern genauso wie die anderen Java–Programme auf dem PC mit der integrierten Entwicklungsumgebung Eclipse. Dazu benötigen wir das Android SDK sowie ein Zusatzmodul für Eclipse, das Plugin für Android[1]. Das Android SDK enthält mit dem AVD[2] einen Simulator, mit dessen Hilfe wir Apps auch auf dem PC laufen lassen können. **Abbildung 1.1** zeigt die prinzipiellen Abläufe. Eine App besteht nicht nur aus einem Pro-

[1] Die Schritte zur Installation finden Sie im nächsten Abschnitt.
[2] Android Virtual Device

gramm, sondern enthält noch sog. Ressourcen wie Bilder, Texte oder Beschreibungen im `.xml`–Format. Außerdem muss die App alle verlangten Zugriffe, z. B. auf die Kontakte oder auf die SD–Karte des Smartphones benennen.

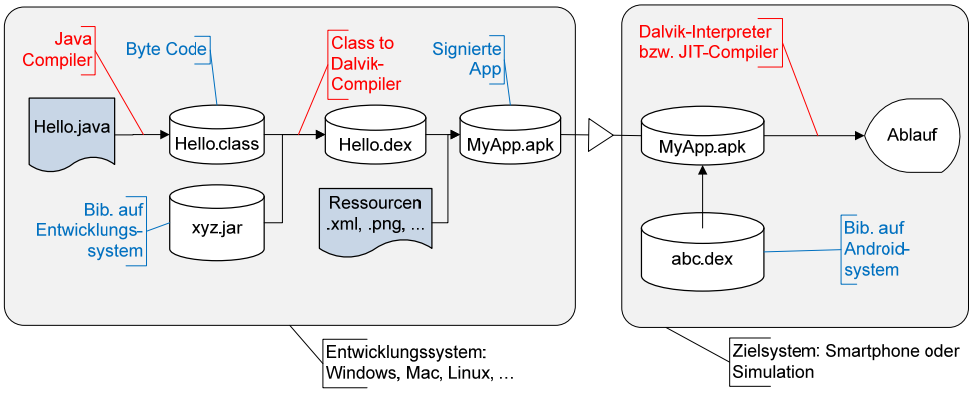

Abbildung 1.1 Entwicklung von Apps für Android

Der Java–Compiler übersetzt das Java–Programm für die App wie alle anderen Java–Programme in eine `.class`–Datei. Diese `.class`–Datei ist nun im Gegensatz zu Plattformen wie der Java–Micro–Edition nicht direkt auf dem Smartphone ablauffähig, sondern wird von einem Compiler in das Dalvík Executable Format `.dex` umgewandelt. Dan Bornstein hat dieses Format erfunden, um besonders kurze und effiziente Programme zu erzeugen, damit das Smartphone Strom sparen kann. Dalvík ist eine kleine Stadt in Island, zu der Dan Bornstein enge Verbindungen hat. Die Entwicklungsumgebung schnürt nun aus dem Programm sowie seinen Ressourcen ein `.apk`–Paket und versieht es mit einer Prüfsumme, der sog. digitalen Signatur. Diese Signatur kann nicht verfälscht werden. Damit kann das Smartphone als Zielsystem die Prüfsumme neu berechnen und mit dem in der App angegebenen Wert vergleichen und so überprüfen, ob die App von irgendwelchen Viren verändert wurde.

Von diesen Abläufen merkt der Benutzer der Eclipse–Entwicklungsumgebung nichts, man bringt seine App wie ein übliches Java–Programm zum Ablauf. Dazu kann man das Smartphone über ein USB–Kabel an den PC anschließen. Dann besorgt das Entwicklungssystem alle Schritte: Übersetzen, Packen, Signieren, auf das Smartphone kopieren, dort installieren und auch gleich zum Ablauf zu bringen. Wenn man kein Smartphone für Android hat, benützt man den Simulator.

1.2 Installation der Entwicklungsumgebung für Android

Wie im Buch unter Abschnitt 1.3 beschrieben, benötigen Sie das Java Development Kit JDK sowie eine Eclipse–Variante für die Entwicklung von Java–Programmen. Zusätzlich

müssen Sie das SDK für Android sowie das ADT[3] Plugin für Eclipse installieren. Sie finden alle Schritte unter http://developer.android.com/sdk/installing.html beschrieben. Zur Installation empfiehlt es sich, in der in Tabelle 1.1 beschriebenen Reihenfolge vorzugehen.

Komponente	Download bzw. Beschreibung
JDK	http://www.oracle.com/technetwork/java/javase/downloads/index.html
Eclipse	http://www.eclipse.org/downloads/
SDK für Android	http://developer.android.com/sdk/index.html
ADT Plugin (Beschreibung)	http://developer.android.com/sdk/eclipse-adt.html
ADT-URL für Eclipse	https://dl-ssl.google.com/android/eclipse/

Tabelle 1.1 Werkzeuge zur Entwicklung von Apps für Android

Die Qual der Wahl der Versionen und Varianten der Komponenten

Bei Java empfiehlt sich die jeweils aktuelle Version 1.6, bei Eclipse eine Variante wie z. B. „Eclipse IDE for Java Developers“. Bei Android sind diverse Versionen im Einsatz. Besonders weit ist die Version 2.3 verbreitet, die wesentlich kompletter als die Version 2.2 ausgestattet ist. Achten Sie auf die Version von Android auf Ihrem Smartphone und wählen Sie bei der Installation des SDK für Android eine Version des API, die auf Ihrem Smartphone lauffähig ist. Nach dem SDK müssen wir das ADT Plugin von Google in Eclipse installieren. Folgen Sie dabei der in Tabelle 1.1 angegebenen Beschreibung.

Smartphone über USB anschließen und Apps laufen lassen

Wir erstellen die Programme am PC und lassen sie auf dem Smartphone laufen. Zur Verbindung zwischen PC und Smartphone benützen wir ein USB–Kabel. Unter Windows benötigen Sie einen für das Smartphone geeigneten USB–Treiber. Bei Smartphones der Firma Samsung ist u. U. die Installation des Dienstprogramms „Kies“ der Firma Samsung erforderlich. Bei der Installation von „Kies“ werden die benötigten Treiber für den USB mit installiert. Siehe [http://developer.android.com/sdk/installing.html]

Die Installation von Apps auf Smartphones ist ein sicherheitskritischer Schritt und deswegen nicht ohne die ausdrückliche Erlaubnis möglich. Dies muss man unter Android 4.x unter Einstellungen/Entwickleroptionen/USB-Debugging explizit erlauben, unter älteren Android Versionen unter Einstellungen/….

Vorbereiten von Eclipse

Abbildung 1.2 zeigt die für die Entwicklung unter Android möglichen Views, die man aus der Eclipse–Entwicklungsumgebung unter *Window/Show View/Other...* aktivieren kann. Die View *Devices* zeigt die angeschlossenen Geräte. Drückt man in dieser Ansicht das Symbol für Kamera, so erhält man Bildschirmabdrucke vom über USB angeschlossenen

[3] Android Development Tools

Smartphone. Dies ist auch unabhängig vom Programmieren nützlich. Die View *LogCat* liefert eine Ansicht auf alle Systemmeldungen. Damit kann man Ausgaben des Programms auf dem Smartphone lesen. Dieses sog. *Logging* kann bei der Suche nach Fehlern sehr hilfreich sein. Mit der View *File Explorer* haben Sie Zugriff auf einen Teil des Dateisystems des Smartphones. Sie können auf die das Dateisystem der SD-Karte zugreifen und Dateien vom PC auf das Smartphone und umgekehrt übertragen.

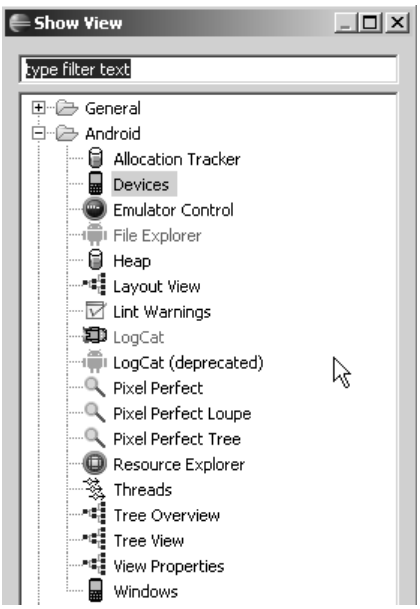

Abbildung 1.2 Neue Ansichten für Android auswählen: Devices, File Explorer, LogCat, …

Entwicklungssystem für Android über Eclipse verwalten

Wenn das ADT Plugin installiert ist, können Sie das Entwicklungssystem für Android unter *Window/Android SDK Manager* verwalten. Sie können Teile entfernen oder aktualisisieren. Achten Sie darauf, dass Sie zumindest diejenigen Teile installieren, die Sie für einen Ablauf auf Ihrem Smartphone benötigen.

Unter *Window/AVD Manager* verwalten Sie die sog. *Android Virtual Devices*. Hier legt man Simulatoren für die eigenen Apps an ,wenn man sie nicht auf dem Smartphone laufen lassen will bzw. kann. Der Start eines sog. AVD kann relativ lange dauern. Man braucht etwas Geduld, bis sich der Simulator zeigt. Deswegen sollte man den Simulator nicht nach jedem Programmlauf einer App beenden. Der Simulator bildet das reale Smartphone nicht vollständig ab, es fehlen diverse Sensoren, die Kamera usw. für den Einstieg in die Programmierung mit Android reicht der Simulator aus.

1.3 Erstellung und Ablauf des ersten Programms

Nach dem Start fragt Eclipse nach dem Ordner für den Arbeitsbereich zum Ablegen aller erstellten Programme. Sie können den Ordner bei den eigenen Daten oder in einem beliebigen Verzeichnis anlegen. Wenn man bei zukünftigen Starts von Eclipse Fragen nach der Lage des Arbeitsbereichs vermeiden will, kann man das Häkchen bei der *Use this as default…* –Option anklicken.

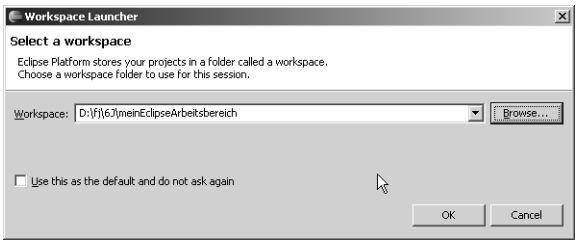

Abbildung 1.3 Auswahl eines Arbeitsbereichs

Danach meldet sich Eclipse mit seinem Willkommensbildschirm. Sie können jetzt das Eclipse–System erkunden oder gleich weiter zum Arbeitsbereich gehen, indem Sie auf das Pfeil–Symbol am rechten Rand des Bildschirms klicken. Danach sehen Sie die Arbeitsoberfläche von Eclipse aus **Abbildung 1.4**. Am linken Rand finden Sie den „Package Explorer“, mit dem Sie die später die einzelnen Projekte erkunden können.

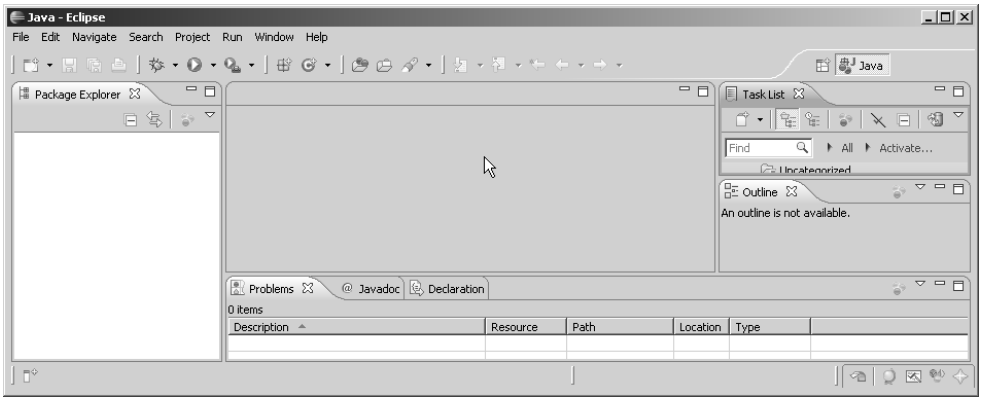

Abbildung 1.4 Arbeitsoberfläche von Eclipse

Zunächst müssen wir unter dem Menüpunkt *File/New/Android Project* ein neues Projekt für Android erzeugen. Das Projekt enthält die Java–Programme der App sowie Bilder und sonstige benötigte Ressourcen. Wir geben in **Abbildung 1.5** einen Namen für das Projekt ein und Klicken auf *Next*. Im nächsten Schritt wählen wir eine Plattform für den Ablauf aus. **Abbildung 1.6** zeigt, wie man aus den installierten Plattformen als Ziel Android 4.0.3 auswählt. Wenn Sie die Programme auf einem Smartphone ablaufen lassen wollen, müssen Sie die Version von Android auf Ihrem Handy oder eine niedrigere Version wählen.

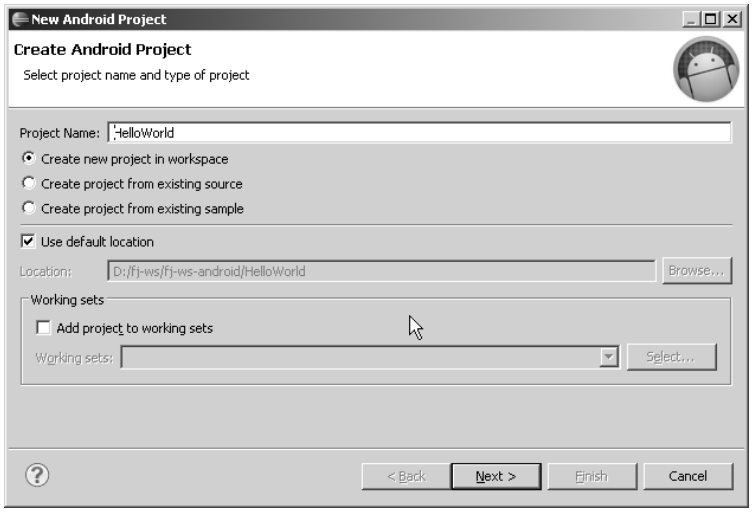

Abbildung 1.5 Der erste Schritt beim Anlegen eines Projekts für Android

Mit *Next* erhalten wir den in **Abbildung 1.7** angegebenen Dialog, in dem wir noch den Namen des Packages sowie den Namen der Activity eingeben müssen.

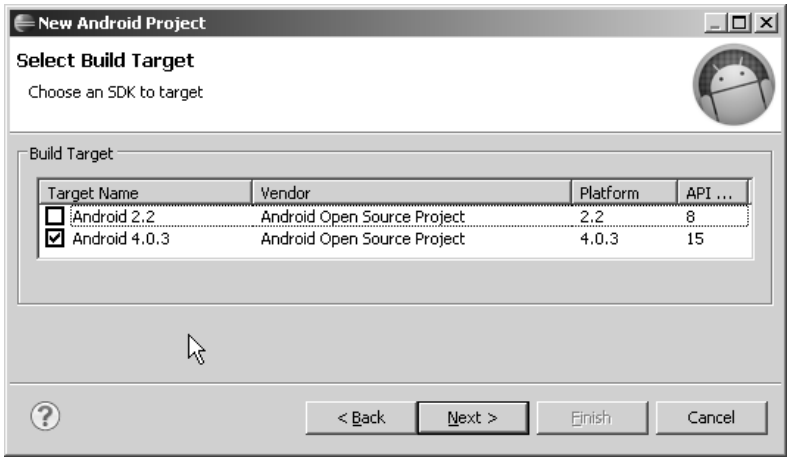

Abbildung 1.6 Der 2. Schritt beim Anlegen eines Projekts für Android: Auswahl einer Plattform

Nach Drücken von *Finish* in **Abbildung 1.7** erhält man in **Abbildung 1.8** eine erste lauffähige App für Android mit einer Activity namens `HelloWorldActivity`.

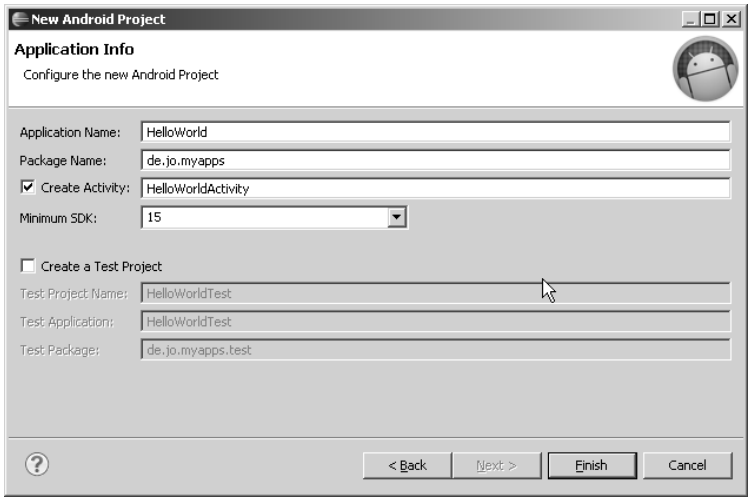

Abbildung 1.7 Der 3. Schritt beim Anlegen eines Projekts für Android

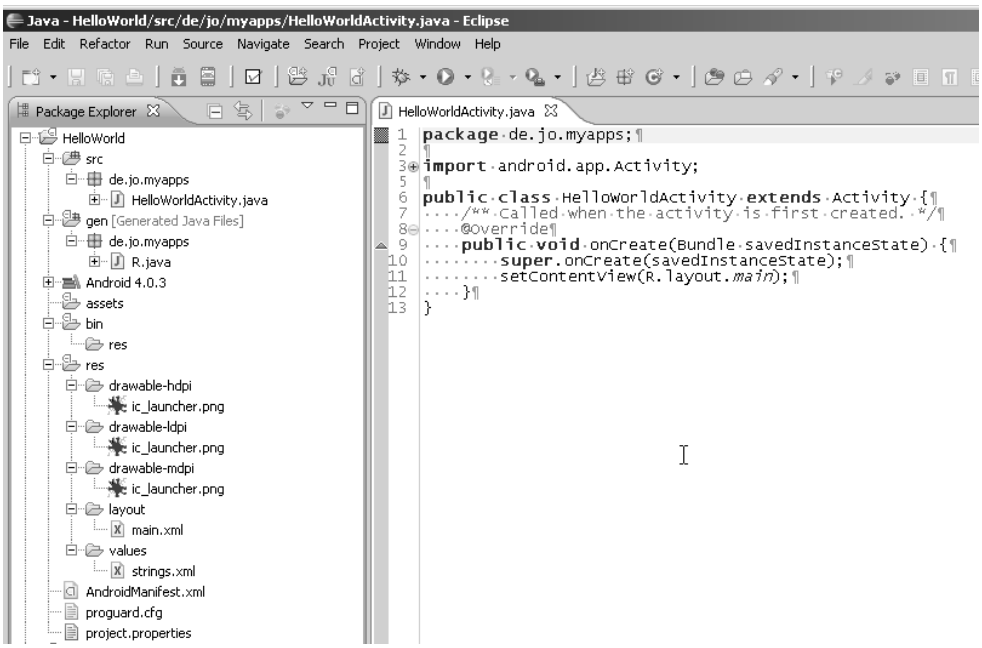

Abbildung 1.8 Die erste App unter Android

Unsere erste App enthält eine Activity `de.jo.myapps.HelloWorldActivity` im Verzeichnis *src*. Die erste Activity benützt die Konstante `R.layout.main`. Im Verzeichnis *gen* finden wir eine Datei `de.jo.myapps.R.java`. Diese Datei wird vom Entwicklungssystem für Android generiert und bei jeder Änderung der sog. Ressourcen neu erzeugt. Das Verzeichnis *res* enthält die Ressourcen: Bilder in verschiedener Auflösung in den Verzeichnis-

sen `drawable?.dpi` (?=l, m, h, xh), das Layout der Activity in der Datei `main.xml` sowie die Datei `values.xml` mit Definitionen von Zeichenketten.

Eclipse übersetzt das Programm ohne besondere Aufforderung durch den Anwender. Nach einer Sicherung des Programms im Arbeitsbereich über das File–Menü oder mit der Tastenkombination *Strg+S* können Sie das Programm mit *Strg+F11* als „Android-Application“ laufen lassen und erhalten wie in **Abbildung 1.9** die Ausgabe des ersten Programms.

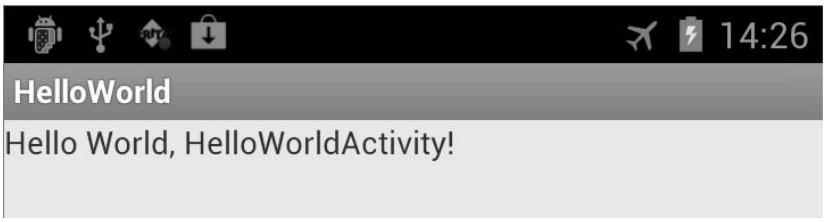

Abbildung 1.9 Unsere erste Android-App läuft!

1.4 Aufbau einer Applikation für Android

Die Applikation für Android besteht aus verschiedenen Teilen:

1. Eine Activity: `HelloWorldActivity.java`
2. Die Datei mit den generierten Konstanten: `R.java`
3. Eine Datei mit der Beschreibung des Layouts: `main.xml`
4. Eine Datei mit Werten: `strings.xml`
5. Die Datei mit den Einstellungen für die App: `AndroidManifest.xml`

Listing 1.1 Die Activity `HelloWorldActivity.java`

```
package de.jo.myapps;

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

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

Unsere Activity ist von der Basisklasse `android.app.Activity` abgeleitet. Die Methode `onCreate (…)` ist überschrieben. Diese Methode wird von Android aufgerufen, wenn das Exemplar der Klasse `HelloWorldActivity` erzeugt ist. Wir rufen zunächst die entsprechende Methode der Basisklasse auf, die dann evtl. vorhandene Komponenten der Benut-

zungsoberfläche wiederherstellt. Danach setzen wir die `View` für das Fenster der Applikation. Die `View` ist die Basisklasse für alle Komponenten grafischer Benutzungsoberflächen unter Android, wobei wir an die Klassen `Component` bzw. `JComponent` von Swing (vgl. Kap. 7) denken können. In **Listing 1.1** wird die `View` aus einer Beschreibung in einer Datei im Ressourcen-Teil der Applikation aufgebaut.

Man bezeichnet `onCreate(…)` als sog. Life Cycle Callback. Jede Activity unter Android hat einen Lebenszyklus (siehe Kap. xx). `onCreate(…)` ist die Methode, die als Erste aufgerufen wird. Hier initialisieren wir unsere Variablen, die Benutzungsoberfläche usw.

Listing 1.2 Die generierte Datei `R.java`

```
/* AUTO-GENERATED FILE.  DO NOT MODIFY.
 *
 * This class was automatically generated by the
 * aapt tool from the resource data it found.  It
 * should not be modified by hand.
 */

package de.jo.myapps;

public final class R {
  public static final class attr {
  }

  public static final class drawable {
    public static final int ic_launcher = 0x7f020000;
  }

  public static final class layout {
    public static final int main = 0x7f030000;
  }

  public static final class string {
    public static final int app_name = 0x7f040001;
    public static final int hello = 0x7f040000;
  }
}
```

Diese Datei enthält Konstanten, die wir im Programm verwenden können. Diese Konstanten werden von einem Programm der Entwicklungsumgebung automatisch aus den Ressourcen erzeugt. In der ersten Activity wird die Konstante `R.layout.main` benützt. Sie sehen auch die Konstanten `R.string.hello` bzw. `R.string.app_name` für Texte. Wenn wir alle Texte in der Datei `string.xml` ablegen, können wir die Internationalisierung von Android erhalten, wenn wir nur die Texte in den jeweiligen Sprachen angeben. Sind die Texte dagegen im Programm „hart“ eincodiert, müssen alle entsprechenden Programmteile geändert werden.

Listing 1.3 Das Layout: main.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="fill_parent"
  android:orientation="vertical" >

  <TextView
    android:layout_width="fill_parent"
```

```
    android:layout_height="wrap_content"
    android:text="@string/hello" />

</LinearLayout>
```

Das Layout ordnet alle Komponenten linear an, d. h. die Komponenten werden der Reihe nach von Oben nach Unten (`vertical`) angeordnet. Dabei füllt das Layout den Bereich seiner Eltern (`parent`) sowohl horizontal wie auch vertikal aus.

Die `TextView` dient der Darstellung von Texten. Sie ist eine `View` und füllt den Bereich der Eltern-Komponente (hier das lineare Layout) in horizontaler Richtung aus. In vertikaler Richtung belegt diese Komponente nur den tatsächlich benötigten Platz. Siehe auch **Abbildung 1.9**. Den Text in diesem Textelement sprechen wir über `@string/hello` an.

Listing 1.4 Die Datei mit den Zeichenketten: string.java

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

  <string name="hello">Hello World, HelloWorldActivity!</string>
  <string name="app_name">HelloWorld</string>

</resources>
```

In dieser Datei fassen wir die Texte im Programm zusammen, wenn wir uns eine einfache Internationalisierung ermöglichen wollen. Im Programm könnten wir einen Text wie folgt ausgeben. *LogCat* zeigt uns diesen Text für die Applikation `de.jo.myapps` an.

```
System.out.println(getResources().getString(R.string.app_name));
```

Listing 1.5 Die Datei mit den Einstellungen für die App: AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="de.jo.myapps"
  android:versionCode="1"
  android:versionName="1.0" >

  <uses-sdk android:minSdkVersion="14" />

  <application
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name"
    android:theme="@android:style/Theme.Light" >
    <activity
      android:name=".HelloWorldActivity"
      android:label="@string/app_name" >
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>
  </application>

</manifest>
```

Die Datei `AndroidManifest.xml` enthält die zentralen Einstellungen für die App. Dort legen wir das Package, die Version sowie die benötigte Version von Android fest. Als einzige Änderung gegenüber der vom Android Plugin generierten Variante sehen Sie das

Android-Thema „Light“: Der Hintergrund ist standardmäßig dunkel bei hellem Vordergrund. Mit der „Light“-Einstellung erscheint der Hintergrund hell und der Vordergrund dunkel. Sie können ein eigenes Icon erstellen und als Icon für die App festlegen.

In dieser Datei führen wir alle von der App angeforderten Rechte auf. Wenn der Anwender bei der Installation der App die angeforderten Rechte wie Zugriff auf das Netzwerk usw. einräumt, dann – und nur dann – darf die App auf das Netzwerk zugreifen.

Wir müssen jede Activity der App in dieser Datei aufführen. Mehr dazu in Kap. xx.

1.5 Aufgaben

Aufgabe 1.1

Legen Sie ein Projekt für Android an und lassen die App unverändert laufen. Dann ändern Sie den Text für die Ausgabe ab und lassen die App wieder laufen. Ändern Sie die Datei `AndroidManifest.xml` wie in **Listing 1.5** beschrieben ab: Setzen Sie das Light-Thema.

Aufgabe 1.2

Fügen Sie den bei **Listing 1.4** angegebenen Befehl zur Ausgabe in die Methode `onCreate(…)` ein. Was beobachten Sie in der Ausgabe von LogCat, wenn Sie die App auf Ihrem Smartphone laufen lassen und dieses um 90° drehen?