Overview

- a complete platform for mobile devices
  - based on Linux
- originally developed by Android, Inc. company
- 2005 – bought by Google
- 2007 – Open Handset Alliance
  - Google, HW and SW developing companies,

- http://developer.android.com/
  - documentation
  - tutorials
  - tools
    - SDK – core tools
    - Android Studio – IDE, based on IntelliJ IDEA
  - ...

Java, summer semester 2020
Java vs. Android

- ...is it Java or not...?
  - yes and no
    - depends on “point of view”

- programs (primarily) developed in Java
- then it is compiled to byte-code (.class)
- the byte-code is compiled to Dalvik byte-code (.dex)
  - different one than Java byte-code
- this byte-code is executed by
  - Dalvik Virtual Machine  $\leq$ Android 4.4
    - different one than the Java Virtual Machine
  - ART Virtual Machine  $\geq$ Android 5
    - different one than the Java Virtual Machine
Java vs. Android

- spring 2016 – change in Android N
  - Jack and Jill tool chain
  - direct compilation from Java to DEX
Java vs. Android

- used from Java
  - language
    - with the same syntax and semantics
  - part of API of std library
Platform structure

source: https://developer.android.com/guide/platform
Note: native applications

• programs can be written also in C/C++
  – it is not a primary way
  – it is necessary to download a separated NDK
    • SDK support only programs in “Java”
  – support of ARM, MIPS and x86 processors
Kotlin & Android

- Kotlin
  - statically typed programming language that runs on the Java virtual machine
  - developed by JetBrains

- 2nd official language for Android development
  - since May 2017
Problem – “fragmentation”

• both software and hardware

• software
  – many still used versions of the system
    • new API
    • deprecated API
    • different recommendation how to develop applications

• hardware
  – hundreds of different devices with Android
    with different features
    • display size, display density, (non)availability of sensors,
     (non)availability of HW buttons,...
## Different versions of Android

<table>
<thead>
<tr>
<th>Android Platform Version</th>
<th>API Level</th>
<th>Cumulative Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.0 Ice Cream Sandwich</td>
<td>15</td>
<td>99.8%</td>
</tr>
<tr>
<td>4.1 Jelly Bean</td>
<td>16</td>
<td>99.2%</td>
</tr>
<tr>
<td>4.2 Jelly Bean</td>
<td>17</td>
<td>98.4%</td>
</tr>
<tr>
<td>4.3 Jelly Bean</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>4.4 KitKat</td>
<td>19</td>
<td>98.1%</td>
</tr>
<tr>
<td>5.0 Lollipop</td>
<td>21</td>
<td>94.1%</td>
</tr>
<tr>
<td>5.1 Lollipop</td>
<td>22</td>
<td>92.3%</td>
</tr>
<tr>
<td>6.0 Marshmallow</td>
<td>23</td>
<td>84.9%</td>
</tr>
<tr>
<td>7.0 Nougat</td>
<td>24</td>
<td>73.7%</td>
</tr>
<tr>
<td>7.1 Nougat</td>
<td>25</td>
<td>66.2%</td>
</tr>
<tr>
<td>8.0 Oreo</td>
<td>26</td>
<td>60.8%</td>
</tr>
<tr>
<td>8.1 Oreo</td>
<td>27</td>
<td>53.5%</td>
</tr>
<tr>
<td>9.0 Pie</td>
<td>28</td>
<td>39.5%</td>
</tr>
<tr>
<td>10. Android 10</td>
<td>29</td>
<td>8.2%</td>
</tr>
</tbody>
</table>
### Different versions of Android (-1y)

<table>
<thead>
<tr>
<th>Version</th>
<th>Codename</th>
<th>API</th>
<th>Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.3.3-2.3.7</td>
<td>Gingerbread</td>
<td>10</td>
<td>0.3%</td>
</tr>
<tr>
<td>4.0.3-4.0.4</td>
<td>Ice Cream Sandwich</td>
<td>15</td>
<td>0.3%</td>
</tr>
<tr>
<td>4.1.x</td>
<td>Jelly Bean</td>
<td>16</td>
<td>1.2%</td>
</tr>
<tr>
<td>4.2.x</td>
<td></td>
<td>17</td>
<td>1.5%</td>
</tr>
<tr>
<td>4.3</td>
<td></td>
<td>18</td>
<td>0.5%</td>
</tr>
<tr>
<td>4.4</td>
<td>KitKat</td>
<td>19</td>
<td>6.9%</td>
</tr>
<tr>
<td>5.0</td>
<td>Lollipop</td>
<td>21</td>
<td>3.0%</td>
</tr>
<tr>
<td>5.1</td>
<td></td>
<td>22</td>
<td>11.5%</td>
</tr>
<tr>
<td>6.0</td>
<td>Marshmallow</td>
<td>23</td>
<td>16.9%</td>
</tr>
<tr>
<td>7.0</td>
<td>Nougat</td>
<td>24</td>
<td>11.4%</td>
</tr>
<tr>
<td>7.1</td>
<td></td>
<td>25</td>
<td>7.8%</td>
</tr>
<tr>
<td>8.0</td>
<td>Oreo</td>
<td>26</td>
<td>12.9%</td>
</tr>
<tr>
<td>8.1</td>
<td></td>
<td>27</td>
<td>15.4%</td>
</tr>
<tr>
<td>9</td>
<td>Pie</td>
<td>28</td>
<td>10.4%</td>
</tr>
</tbody>
</table>

Data for 7.5.2019

### Different versions of Android (-2y)

<table>
<thead>
<tr>
<th>Version</th>
<th>Codename</th>
<th>API</th>
<th>Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.3 -</td>
<td>Gingerbread</td>
<td>10</td>
<td>0.3%</td>
</tr>
<tr>
<td>2.3.7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.0.3 -</td>
<td>Ice Cream Sandwich</td>
<td>15</td>
<td>0.4%</td>
</tr>
<tr>
<td>4.0.4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.1.x</td>
<td>Jelly Bean</td>
<td>16</td>
<td>1.7%</td>
</tr>
<tr>
<td>4.2.x</td>
<td></td>
<td>17</td>
<td>2.2%</td>
</tr>
<tr>
<td>4.3</td>
<td></td>
<td>18</td>
<td>0.6%</td>
</tr>
<tr>
<td>4.4</td>
<td>KitKat</td>
<td>19</td>
<td>10.5%</td>
</tr>
<tr>
<td>5.0</td>
<td>Lollipop</td>
<td>21</td>
<td>4.9%</td>
</tr>
<tr>
<td>5.1</td>
<td></td>
<td>22</td>
<td>18.0%</td>
</tr>
<tr>
<td>6.0</td>
<td>Marshmallow</td>
<td>23</td>
<td>26.0%</td>
</tr>
<tr>
<td>7.0</td>
<td>Nougat</td>
<td>24</td>
<td>23.0%</td>
</tr>
<tr>
<td>7.1</td>
<td></td>
<td>25</td>
<td>7.8%</td>
</tr>
<tr>
<td>8.0</td>
<td>Oreo</td>
<td>26</td>
<td>4.1%</td>
</tr>
<tr>
<td>8.1</td>
<td></td>
<td>27</td>
<td>0.5%</td>
</tr>
</tbody>
</table>

Data for 16. 4. 2018
Different versions of Android

- **minimum SDK version**
  - application property (defined in the manifest)
  - the minimum API Level required for the application to run
    - cannot be installed on lower level devices
  - should be always specified
    - default value = 1

- **target SDK version**
  - the API Level that the application targets
  - the system should not enable any compatibility behaviors to maintain the app's forward-compatibility
  - default value = minSdkVersion

- **maximum SDK version**
  - should be used
    - new Android versions should be always backward-compatible
Different display size/density

- density-independent pixel
  - dp
  - 1dp = 160px/dpi
- images and “views” in multiple variants
  - according to size/density
    - will be discussed later
- 9-patch PNG
  - “stretchable” images
  - .9.png extension
  - a PNG image in which the borders have special meaning
    - left and top – where the image can be stretched
    - right and bottom – content border (e.g. button content)
  - creations – draw9patch tool in SDK
9-patch PNG

images source: http://developer.android.com/training/multiscreen/screensizes.html
Security

- applications run in “sandbox”
- by default, application are allowed to “almost” nothing
- permissions
  - specified in the manifest
  - during application installation, the system shows to a user all required permissions
  - permission examples
    - location (GPS)
    - bluetooth
    - phone function
    - SMS/MMS
    - net access
    - ...

Application structure

- Activities
  - UI components
  - application's entry points
- Views
  - UI elements
- Intents
  - asynchronous messages
- Services
  - long-running services in the background without UI
- Content providers
  - data providers for other applications
- Broadcast Intent Receivers
  - broadcast listeners (e.g. low battery level notifications)
- (HomeScreen) Widgets
  - interactive components on “desktop”
Project creation

- in IDE
  - New project...
- formerly also from the command-line
  - android tool
  - deprecated
Project creation

- project “parameters”
  - Application Name
    - human readable name
  - Package Name
    - “root” package serving as the application identifier
    - naming convention should be held
  - Target (min SDK version)
    - it is not directly the API level
    - command `android list`
      - a list of all supported targets
Project structure

- AndroidManifest.xml
- res/
- src/
Project structure

- AndroidManifest.xml
  - application description
    - components
    - requirements
    - ...

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest ...
  <uses-sdk android:minSdkVersion="8"
    android:targetSdkVersion="17" />
  <application android:icon="@drawable/app_icon.png" ...
    android:title="com.example.project.ExampleActivity"
    android:label="@string/example_label" ...
  
  </activity>
  ...
</application>
<manifest>
Project structure

- **res/** – resources
  - subdirectories in the directory res
    - drawable
      - images
      - ...
    - values
      - strings
      - ...
    - layouts
      - screens
  - the R class
    - generated class
    - contains resource identifiers
      - as static fields
      - these are used in code
Project structure

• resources can have variants
  - specified by extension
  - drawable-hdpi, drawable-ldpi, drawable-mdpi
    • images for high, low and middle density of a display
  - other extension
    • land, port – display orientation
    • cs, en, fr, … – device language
    • small, normal, large – display size
    • ...
  - extensions can be combined
  - př:
    • res/values-de/
    • res/values-cs/
    • res/drawable-cs/
    • res/drawable-en-rUK/
Launching application

• in an emulator
  - IDE – Menu Tools-> AVD manager
• in a real device
  - attached via USB

• compilation
  - `gradlew assembleDebug`
• installation (to emulator/device)
  - `adb install`
    `app/build/outputs/MyFirstApp-debug.apk`
Activity

• extends `android.app.Activity`
• a window of the application
  - can serve as an entry point of the application
    • launcher
• its appearance typically described in an xml file
  - in res/layout
import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;

class HelloAndroid extends Activity { 
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        TextView tv = new TextView(this);
        tv.setText("Hello, Android");
        setContentView(tv);
    }
}

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

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

res/layout/main.xml
<?xml version="1.0" encoding="utf-8"?>
<TextView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/textview"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:text="@string/hello"/>

res/values/strings.xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello, Android!
        I am a string resource!</string>
    <string name="app_name">Hello, Android</string>
</resources>
Activity lifecycle

• similarly to Swing
• a hierarchy of objects
  – children of **View** and **ViewGroup**

source: https://developer.android.com/guide/topics/ui/declaring-layout
**ViewGroup ~ Layout**

- children of ViewGroup
- LinearLayout
  - places elements „in a row“
    - android:orientation="vertical"
    - android:orientation="horizontal"
- RelativeLayout
  - element placement relative to other elements
  - an example on the next slide
- TableLayout
- GridLayout
- TabLayout
- ListView
Relative Layout example

```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <TextView
        android:id="@+id/label"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Type here:"/>
    <EditText
        android:id="@+id/entry"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:background="@android:drawable/editbox_background"
        android:layout_below="@id/label"/>
    <Button
        android:id="@+id/ok"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/entry"
        android:layout_alignParentRight="true"
        android:layout_marginLeft="10dip"
        android:text="OK"/>
    <Button
        android:id="@+id/cancel"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toLeftOf="@id/ok"
        android:layout_alignTop="@id/ok"
        android:text="Cancel"/>
</RelativeLayout>
```
Fragments

• since Android 3.0
  – there is the “support library”, which adds support for older versions (for the API level 4 and higher)
    • beware of the package
      \texttt{android.app.Fragment}
      \texttt{android.support.v4.app.Fragment}
• a reusable part of user interface
  ~ an “inner activity” with own layout and life-cycle
• an activity can show several fragments
• easy creation of UI for different types of display
  – phone
  – tablet
Using fragments

source: http://developer.android.com/training/basics/fragments/fragment-ui.html
Using fragmentů

- **fragment**
  public class ArticleFragment extends Fragment {
    @Override
    public View onCreateView(LayoutInflater inflater,
                            ViewGroup container, Bundle savedInstanceState) {
      return inflater.inflate(R.layout.article_view,
                             container, false);
    }
  }

- **res/layout-large/news_articles.xml:**
  <LinearLayout xmlns:android="....."
                android:orientation="horizontal"
                android:layout_width="fill_parent"
                android:layout_height="fill_parent">
    <fragment android:name="HeadlinesFragment"
             android:id="@+id/headlines_fragment"
             android:layout_weight="1"
             android:layout_width="0dp"
             android:layout_height="match_parent" />
    <fragment android:name="ArticleFragment" .... />
  </LinearLayout>
Using fragments

- activity
  ```java
  public class MainActivity extends FragmentActivity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.news_articles);
    }
  }
  ```

- if the min API level is at least 11, the regular `Activity` can be used
Using fragments

- the previous example – fixed UI with two fragments suitable e.g. for a tablet
  - note the large extension of the layout

- for switching fragments (e.g. on a phone) it is necessary to manipulate fragments from code

- res/layout/news_articles.xml

  `<FrameLayout xmlns:android="..."
    android:id="@+id/fragment_container"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />

- empty layout – content is added from code
- without the large extension, i.e. for other display sizes
public class MainActivity extends FragmentActivity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.news_articles);
        if (findViewById(R.id.fragment_container) != null) {
            if (savedInstanceState != null) {
                return;
            }
            HeadlinesFragment firstFragment = new HeadlinesFragment();
            firstFragment.setArguments(getIntent().getExtras());
            getSupportFragmentManager().beginTransaction()
                .add(R.id.fragment_container, firstFragment).commit();
        }
    }
}
Using fragments

- replacing the shown fragment

```java
ArticleFragment newFragment = new ArticleFragment();
FragmentTransaction transaction =
    getSupportFragmentManager().beginTransaction();
transaction.replace(R.id.fragment_container,
    newFragment);
transaction.addToBackStack(null);
transaction.commit();
```
Intents

- application components (activities, services, broadcast receivers) are activates by Intents
  - “messages”
  - Intent – a passive object
    - extends android.content.Intent
    - properties
      - component name
      - action
        - string
          - many predefined
          - own ones can be created
      - data
        - URI of data to work with
      - category
        - other information about component type, which should react to the intent
      - extras
      - flags
Intents

- **explicit**
  - with a name of the target component
  - typically used inside an application

- **implicit**
  - without a component name
  - typically communication between applications

- **intent filters**
  - which intents the component can serve
  - declared in the manifest

```xml
<intent-filter>
  <action android:name="android.intent.action.MAIN" />
  <category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
```


**Intents**

- a permission to react to an intent can be set
  - declared in the manifest
  - confirmed during installation
- also the “system” applications react to intents
  - > own “system” applications can be developed
    - Mailer, SMS app, Homepage,...
private static final int ACTIVITY_PICK_CONTACT = 42;
private void pickContact() {
    Intent intent = new Intent(Intent.ACTION_PICK,
                               ContactsContract.Contacts.CONTENT_URI);
    startActivityForResult(intent, ACTIVITY_PICK_CONTACT);
}

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    switch (requestCode) {
    case (ACTIVITY_PICK_CONTACT) :
        if (resultCode == Activity.RESULT_OK) {
            Uri pickedContact = data.getData();
            return;
        }
        break;
    }
}
Task

• a stack of launched activities
  - an activity reacts to an intent = a new instance is created and put to a stack
• a user communicates with an activity on the top
• several parallel tasks can exist

• task ~ running application
Services

• background running services
• potomci od android.app.Service
  – they do not automatically start their thread!

• IntentService
  – extends Service
  – intended for services reacting to intents
  – they contain thread management
  – it is enough to override void onHandleIntent(Intent intent)
Threads

• activities of an application are run in one thread
• events are also served in this thread
  – “main” thread / UI thread
• similarly as in Swing

• UI is not “thread-safe”
  – manipulations with UI perform in the “main” thread
  – the “main” thread should not be blocked

• helper methods
  – Activity.runOnUiThread(Runnable)
  – View.post(Runnable)
  – View.postDelayed(Runnable, long)

• AsyncTask
  – similar to SwingWorker
public class ADialogFragment extends DialogFragment {
    @Override
    public Dialog onCreateDialog(Bundle savedInstanceState) {
        AlertDialog.Builder builder =
            new AlertDialog.Builder(getActivity());
        builder.setMessage("message")
            .setPositiveButton("OK",
                new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int id) {
                        
                    }
                })
            .setNegativeButton("Cancel",
                new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int id) {
                        
                    }
                });
        return builder.create();   }   }
Dialogs

• showing a dialog
  `ADialogFragment aDialog = new ContactDialogFragment();
  aDialog.show(getFragmentManager(), "dialog");`
@Override
protected Dialog onCreateDialog(int id) {
    switch (id) {
    case DIALOG_SHOW_CONTACT: {
        return new AlertDialog.Builder(this).
            setTitle("XXX").setMessage("Message").setCancelable(true).
            setPositiveButton("OK", null).create();
    }
    }
    return null;
}

@Override
protected void onPrepareDialog(int id, Dialog dialog) {
    switch (id) {
    case DIALOG_SHOW_CONTACT: {
        if (pickedContact != null) {
            ((AlertDialog) dialog).setMessage("YYY");
        }
    }
    }
}

Dialogs – deprecated way

- showDialog(DIALOG_SHOW_CONTACT);
  - showing a dialog