

CS 213 – Software Methodology Spring 2017

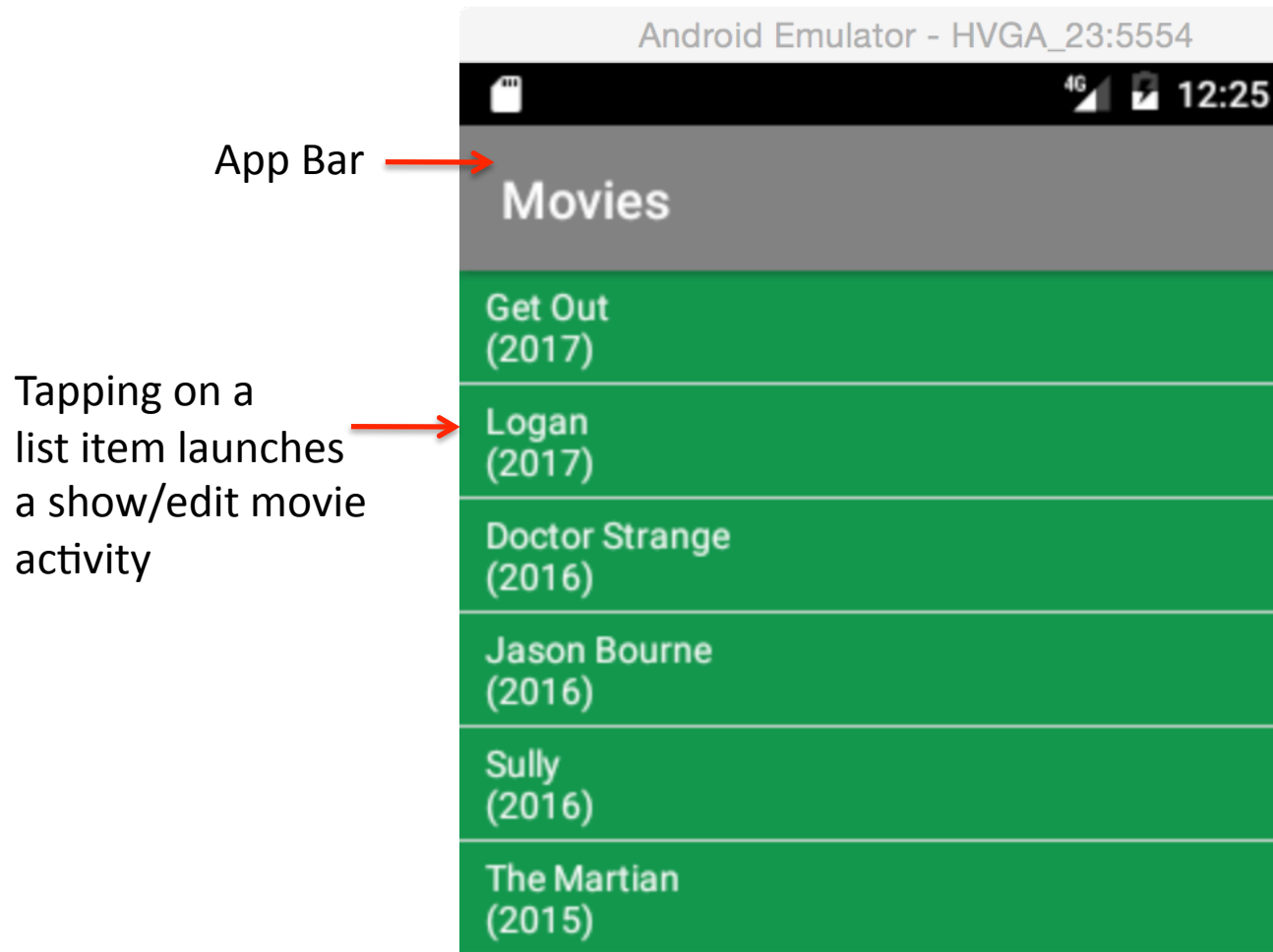
Sesh Venugopal

Lecture 21 – Apr 11
Android Programming

Activities/Dialogs/Icons/IO with non-raw files

Movies List Project

List set up like in the Rutgers Bus Routes app



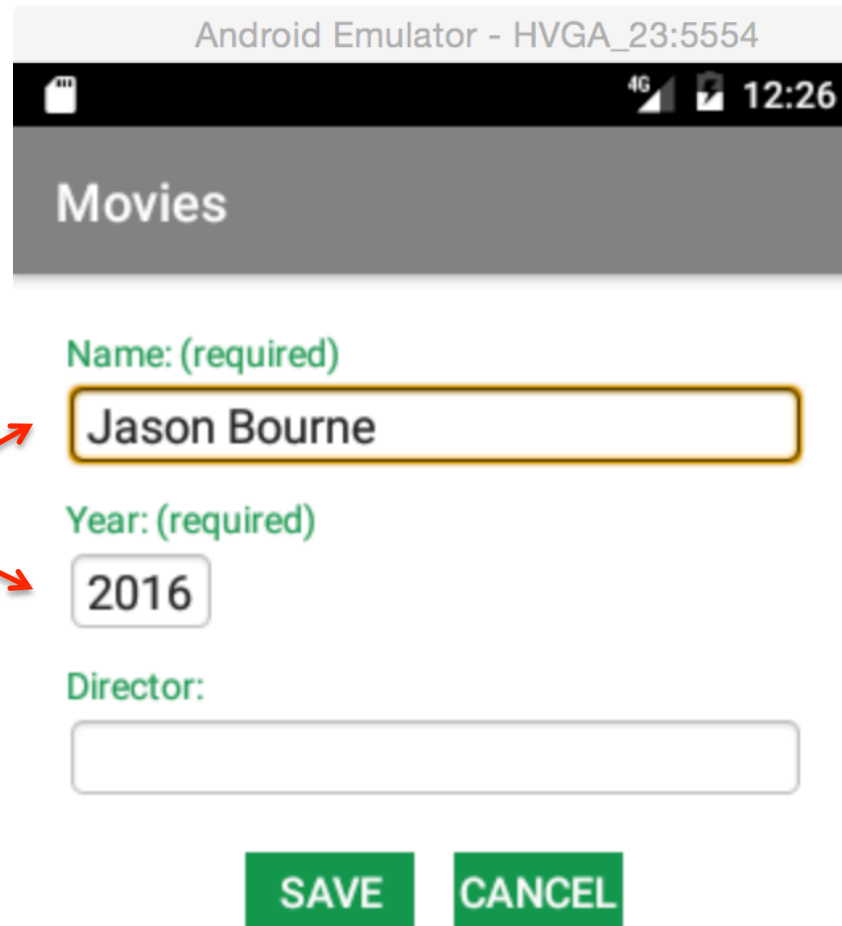
Part 1:

Launching Activity for Result

AddEditMovie Activity

This activity sets up text fields that are pre-populated if it is launched for Show/Edit

A movie item was
tapped in the parent
movie list activity



The screenshot shows an Android emulator window titled "Android Emulator - HVGA_23:5554". The status bar at the top displays a 4G signal icon and the time 12:26. The app's title bar is grey and contains the word "Movies". Below the title bar, the form has three fields: "Name: (required)" with the value "Jason Bourne", "Year: (required)" with the value "2016", and "Director:" with an empty field. At the bottom, there are two green buttons labeled "SAVE" and "CANCEL". Two red arrows point from the text "A movie item was tapped in the parent movie list activity" to the "Name" and "Year" fields.

Name: (required)
Jason Bourne

Year: (required)
2016

Director:

SAVE CANCEL

Launching Activity for Result

The Show/Add/Edit activity might return a result (when Save button is clicked) that needs to be communicated back to the launching parent

`Movies.java`

```
public static final int EDIT_MOVIE_CODE=1;
```

```
...  
public void showMovie(int pos) {  
    Bundle bundle = new Bundle();  
    Movie movie = movies.get(pos);  
    bundle.putInt(AddEditMovie.MOVIE_INDEX,pos);  
    bundle.putString(AddEditMovie.MOVIE_NAME,movie.name);  
    bundle.putString(AddEditMovie.MOVIE_YEAR,movie.year);  
    bundle.putString(AddEditMovie.MOVIE_DIRECTOR,movie.director);  
    Intent intent = new Intent(this, AddEditMovie.class);  
    startActivityForResult(intent, EDIT_MOVIE_CODE);  
}
```

Bundle for movie info to send
to child activity for display

An activity could launch several
children activities with different
request codes



So when a child activity returns, this
code will be used to determine which
of potentially several children
activities it is

AddEditMovie Class

Set up with constants and instance fields:

```
public class AddEditMovie  
extends AppCompatActivity {
```

Keys used to ship info from and
to the parent **Movies** activity

```
    public static final String MOVIE_INDEX = "movieIndex";  
    public static final String MOVIE_NAME = "movieName";  
    public static final String MOVIE_YEAR = "movieYear";  
    public static final String MOVIE_DIRECTOR = "movieDirector";
```

```
    private int movieIndex; ← The position of the movie in the array list of  
                             movies in the Movies activity
```

```
    private EditText movieName, movieYear, movieDirector;
```

```
    ...
```

```
}
```

↑
Text fields in fill out form

AddEditMovie Class

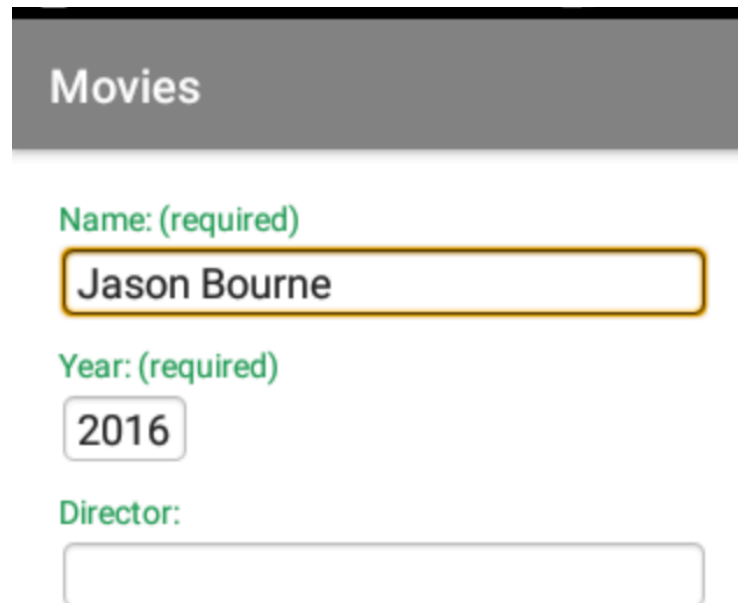
If the incoming **Bundle** is not null (if called to Show/Edit), then get info and populate fields

```
public class AddEditMovie
extends AppCompatActivity {
    ...
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        ...
        // get the fields
        movieName = (EditText)findViewById(R.id.movie_name);
        movieYear = (EditText)findViewById(R.id.movie_year);
        movieDirector = (EditText)findViewById(R.id.movie_director);

        // see if info was passed in to populate fields
        Bundle bundle = getIntent().getExtras();
        if (bundle != null) {
            movieIndex = bundle.getInt(MOVIE_INDEX);
            movieName.setText(bundle.getString(MOVIE_NAME));
            movieYear.setText(bundle.getString(MOVIE_YEAR));
            movieDirector.setText(bundle.getString(MOVIE_DIRECTOR));
        }
    }
}
```

AddEditMovie Event Handling

Save and Cancel buttons are fitted with respective event handling methods in `AddEditMovie` class



A screenshot of a mobile application form titled "Movies". The form contains three input fields: "Name: (required)" with the text "Jason Bourne", "Year: (required)" with the value "2016", and "Director:" which is empty. The form is styled with a grey header bar and green text for labels.

In `add_edit_movie.xml`
event handler:
`android:onClick="save"`



Method name

In `add_edit_movie.xml`
event handler:
`android:onClick="cancel"`



Method name

AddEditMovie Event Handling

Handling **Cancel** event in **AddEditMovie** class

```
public void cancel(View view) {  
    setResult(RESULT_CANCELED);  
    finish();  
}
```

Button that was clicked

Calling this method
results in termination of
activity, with a return to
the previous activity on
call stack

Result code that is sent back to parent
activity, code is a constant defined in
the **AppCompatActivity** class (of
which **AddEditMovie** is a subclass)

AddEditMovie Event Handling

Handling **Save** event in **AddEditMovie** class

```
public void save(View view) {  
    // gather all data from text fields  
    String name = movieName.getText().toString();  
    ...  
    // pop up dialog if errors in input, and return  
    ...  
    // make Bundle  
    Bundle bundle = new Bundle();  
    bundle.putInt(MOVIE_INDEX, movieIndex);  
    bundle.putString(MOVIE_NAME, name);  
    bundle.putString(MOVIE_YEAR, year);  
    bundle.putString(MOVIE_DIRECTOR, director);  
  
    // send back to caller  
    Intent intent = new Intent();  
    intent.putExtras(bundle);  
    setResult(RESULT_OK, intent);  
  
    finish(); // pops activity from the call stack, returns to parent  
}
```



Mechanism to send result
back to parent activity

AddEditMovie finishes

- When `AddEditMovie` finishes, execution returns to `Movies` via a call to its `onActivityResult` method (callback)

`@Override`

```
protected void onActivityResult(int requestCode,  
                                int resultCode,  
                                Intent intent) {
```

```
    if (resultCode != RESULT_OK) { return; }
```

```
    Bundle bundle = intent.getExtras();  
    if (bundle == null) { return; }
```

```
    // gather all info passed back by launched activity  
    String name = bundle.getString(AddEditMovie.MOVIE_NAME);  
    String year = bundle.getString(AddEditMovie.MOVIE_YEAR);  
    String director = bundle.getString(AddEditMovie.MOVIE_DIRECTOR);  
    int index = bundle.getInt(AddEditMovie.MOVIE_INDEX);
```

```
    ...
```

```
}
```

Code that was sent
in by `Movies` when
`AddEditMovie` was
launched

Code returned by
`AddEditMovie`

Intent returned by
`AddEditMovie`

Only relevant if `AddEditMovie` was called for
Show/Edit (this was sent to `AddEditMovie`, and is
passed back)

AddEditMovie finishes

- `onActivityResult` method distinguishes between return from Show/Edit and Add

```
@Override
protected void onActivityResult(int requestCode,
                                int resultCode,
                                Intent intent) {

    ...

    Movie movie = movies.get(index);
    movie.name = name;
    movie.year = year;
    movie.director = director;

    // redo the adapter to reflect change
    listView.setAdapter(
        new ArrayAdapter<Movie>(this,
                                R.layout.movie, movies));

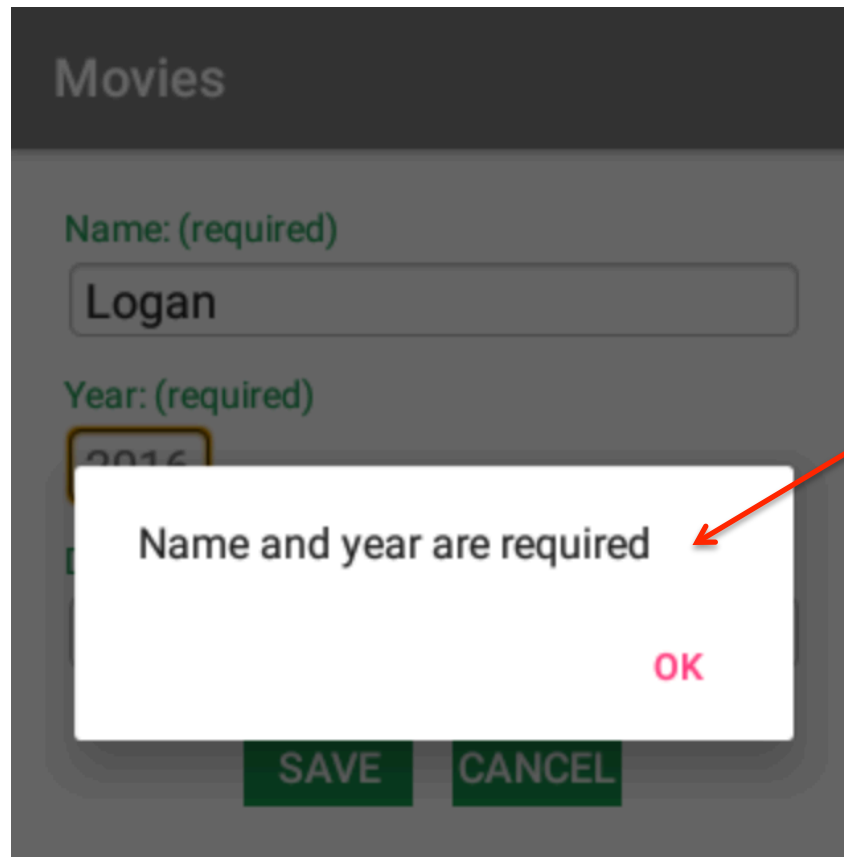
    ...
}
```

Adapter has to be redone
since source content has changed

Part 2: Dialogs

Popping up an error dialog

If either the movie name or year is missing, a dialog is popped up



Dialog is a subclass of
`android.app.AlertDialog`

Popping up an error dialog

See <https://developer.android.com/guide/topics/ui/dialogs.html>

From the package `android.app`

`public class MovieDialogFragment extends DialogFragment {`
 `public static final String MESSAGE_KEY = "message_key";`

The actual text to be shown can be passed in when the fragment instance is created, as value for this key

```
@Override
public Dialog onCreateDialog(Bundle savedInstanceState) {
    ... // create the dialog
}
}
```

DialogFragment onCreateDialog

The `onCreate` method of `DialogFragment` creates an `AlertDialog`, using a standard suggested coding process:

```
@Override
public Dialog onCreateDialog(Bundle savedInstanceState) {
    // Use the Builder class for convenient dialog construction
    Bundle bundle = getArguments();
    AlertDialog.Builder builder =
        new AlertDialog.Builder(getActivity());
    builder.setMessage(bundle.getString(MESSAGE_KEY))
        .setPositiveButton("OK",
            new DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog,
                    int id) {
                    // NOTHING TO DO
                }
            });
    // Create the AlertDialog object and return it
    return builder.create();
}
```

setNegativeButton would allow us to set up a Cancel button, which we don't need here since it's just an info dialog

message sent in when Fragment is created

Showing the dialog with required message

The `save` method of `AddEditMovie` checks if required fields are filled, and if not, creates and shows an instance of the `MovieDialogFragment`:

```
public void save(View view) {  
    ...  
    // name and year are mandatory  
    if (name == null || name.length() == 0 ||  
        year == null || year.length() == 0) {  
  
        Bundle bundle = new Bundle();  
        bundle.putString(MovieDialogFragment.MESSAGE_KEY,  
                        "Name and year are required");  
  
        DialogFragment newFragment = new MovieDialogFragment();  
        newFragment.setArguments(bundle);  
        newFragment.show(getFragmentManager(), "badfields");  
  
        return; // does not quit activity, just returns from method  
    }  
    ...  
}
```

Unique tag name for Fragment,
can be used to get a handle to
the Fragment with
`getFragmentManager().findFragmentById(R.id.fragment_id)`

Part 3:

Getting Icons

Using an Icon for Adding Movie

- We will use a '+' icon to add movies. This icon will show up as an item in the Action/App Bar
- There are prefab icons supplied by the Android guys for a whole lot of standard tasks, including one to add content (such as songs in our app)

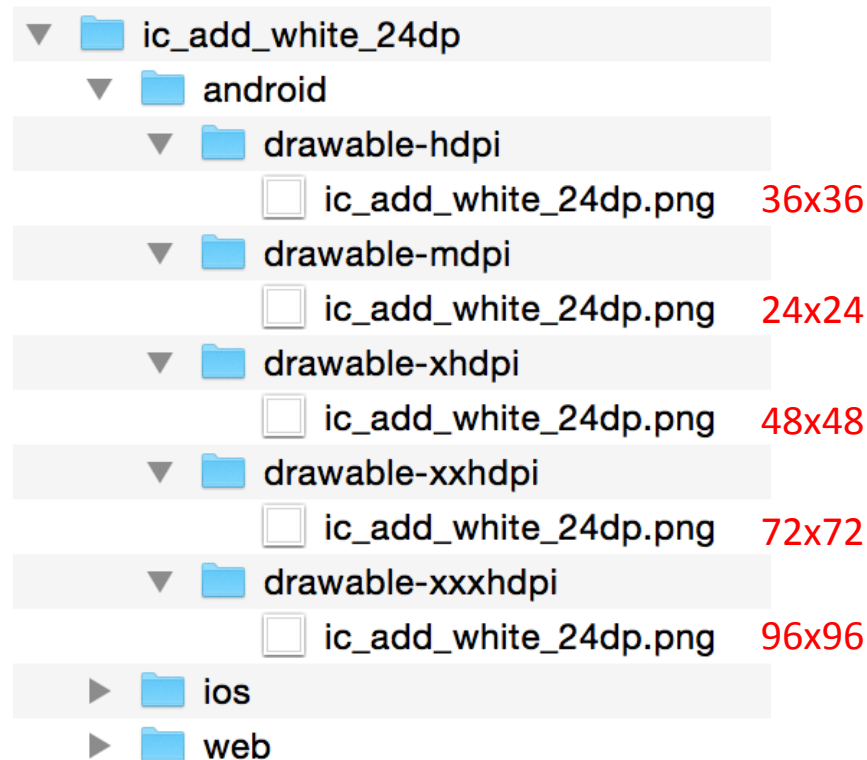
Go to [Material Icons \(https://material.io/icons\)](https://material.io/icons)

On this page, search for “Content” – this will show a collection of Content icons. The first one of them is the '+'

Click on the '+' icon: this brings up a tool bar at the bottom of the browser page. Select (say) the white version and download the PNGs. This will download a zip file which unzips to a folder named [ic_add_white_24dp](#)

+ icon for various screen densities

- The downloaded collection of '+' icons is distributed over several folders, one per screen density, with different sizes



(See [Develop -> API Guides -> Best Practices -> Supporting Multiple Screens](https://developer.android.com/guide/practices/screens_support.html))
https://developer.android.com/guide/practices/screens_support.html

Adding icons to project

Right click on **res**, then choose **New -> Image Asset**, then configure like this

Name that will be used in app

Although the source image is white, if black is needed, just choose HOLO_LIGHT for theme

Asset Studio

Configure Image Asset
Android Studio

Icon Type: Action Bar and Tab Icons

Name: ic_action_plus_white

Asset Type: ☒ Image ☐ Clip Art ☐ Text

Path: le-hdpi/ic_add_white_24dp.png

Trim: ☐ Yes ☒ No

Padding: 0 %

Theme: HOLO_DARK

Source Asset:

xxhdpi xhdpi hdpi mdpi

Cancel Previous Next Finish

Choose any of the icons, and it will automatically scale as needed and drop copies into various **res/drawable** folders one per screen density

(if this doesn't work, copy each version of the image to corresponding folder--e.g. drawable-hdpi-- in res)

Adding icons to project

Right click on `res`, then choose **New -> Image Asset**, then configure like this

Name that will be used in app

Although the source image is white, choosing HOLO_LIGHT for theme will convert it to black

Generate Icons

Configure Image Asset
Android Studio

Action Bar and Tab Icons

Name: `ic_action_plus`

Asset Type: ☒ Image ☐ Clipart ☐ Text

Path: `drawable-hdpi/ic_add_white_24dp.png`

Trim? ☐ Yes ☒ No

Padding: 0 %

Theme: `HOLO_LIGHT`

Source Asset:

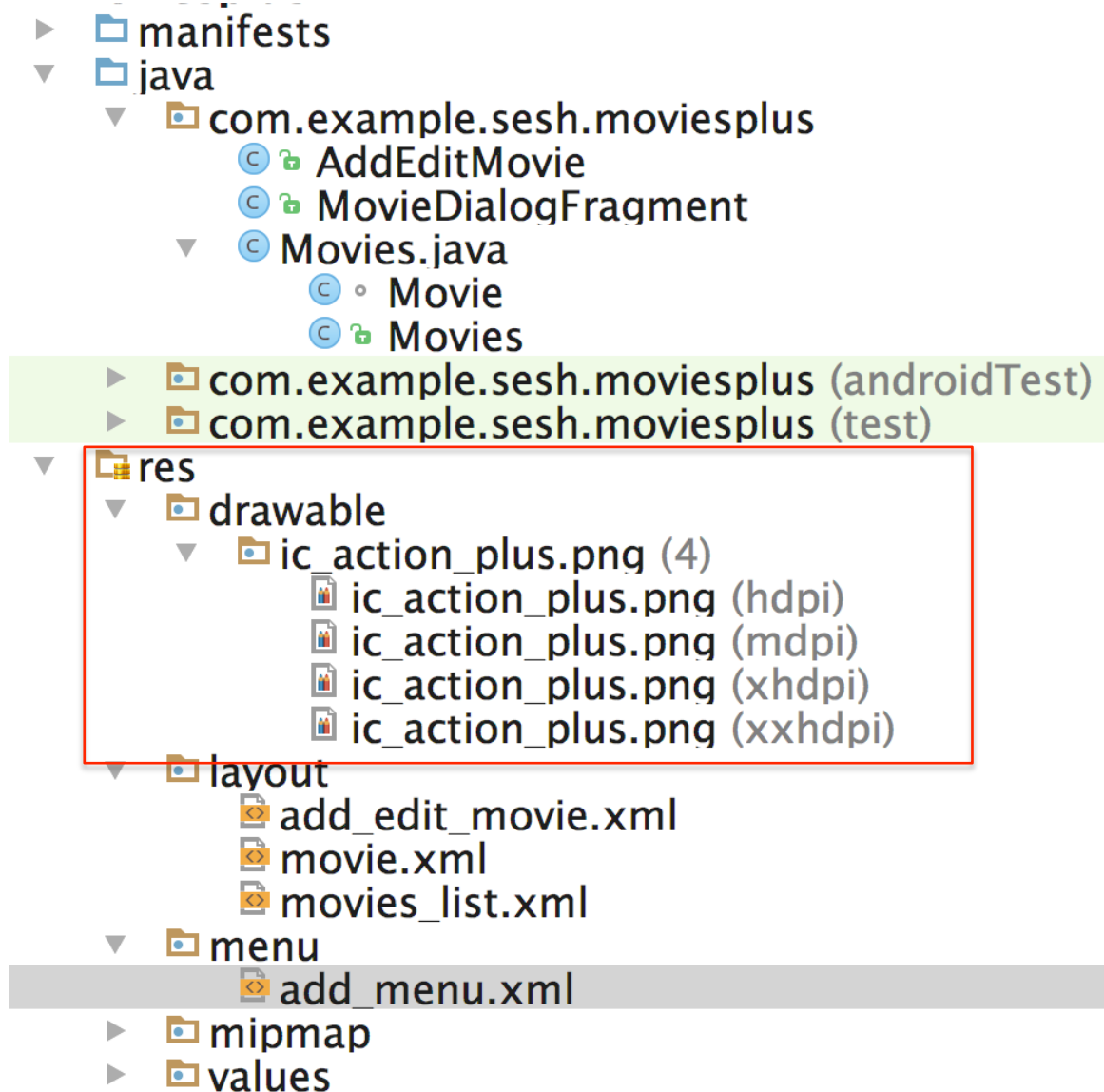
xxhdpi xhdpi hdpi mdpi

Cancel Previous Next Finish

Choose any of the icons, and it will automatically scale as needed and drop copies into various `res/drawable` folders one per screen density

(if this doesn't work, copy each version of the image to corresponding folder--e.g. `drawable-hdpi`-- in `res`)

The various incarnations of ic_action_plus



Part 4:

Adding Icon to Action Bar/Launching AddEditMovie with ADD request code

Adding + icon to Action Bar

- This is a multi-step process:
 - Create a menu resource for the action bar, with the icon as a menu item
 - “Inflate” this menu resource in `Movies.java`, by overriding the callback method that will be invoked when activity is launched, to draw the action bar
 - The menu will not be inflated in `AddEditMovie.java`, so that activity’s action bar will not have the add capability
 - In `Movies.java`, override the method that will be called when a menu item is clicked in the action bar, to handle the add event when + is clicked

Adding +: 1. Create a Menu Resource for Action Bar

- Create a folder called `menu` under `res`
- In the `res/menu` folder, create a menu resource file called `add_menu.xml`, with the following code:

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
      xmlns:appcompat="http://schemas.android.com/apk/res-auto">
  <item android:id="@+id/action_add"
        android:icon="@drawable/ic_action_plus_white"
        android:title="@string/add"
        appcompat:showAsAction="always"/>
</menu>
```

Text version,
in strings.xml

Meaning show at all
times, don't hide it
in overflow menu

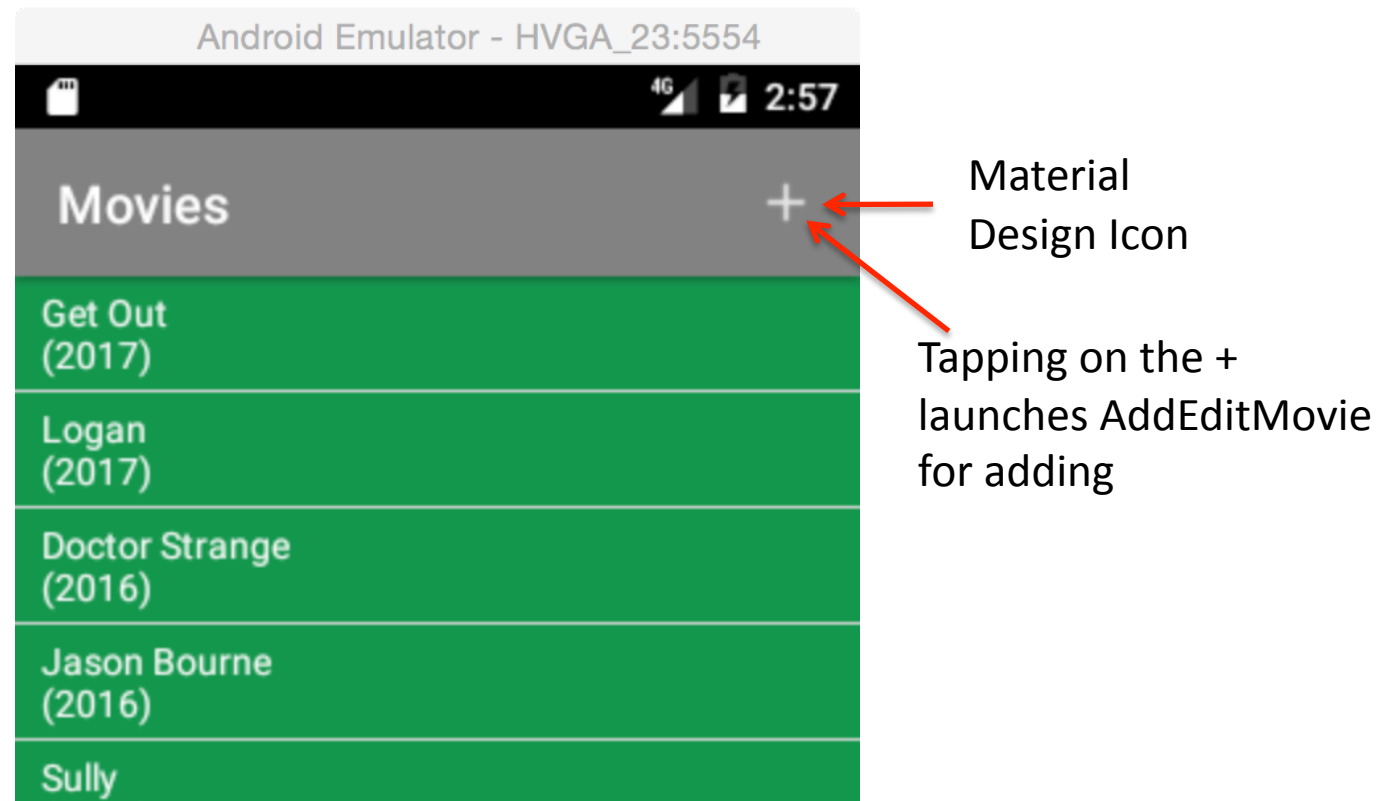
Adding +: 2. Inflate Menu

- In `Movies.java`, override the `onCreateOptionsMenu(Menu)` method to inflate the menu resource – this method will be called when the app is launched

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.add_menu, menu);
    return super.onCreateOptionsMenu(menu);
}
```

Movies List Project

List set up like in the Rutgers Bus Routes app



Show/Edit and Add are done by the same activity, identified by different REQUEST codes sent by this parent activity

Adding +: 3. Override callback for event handling

- In `Movies.java`, override `onOptionsItemSelected` method (which is called whenever an item is clicked in the Action Bar):

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.action_add: ← Id of + icon in menu resource xml
            addMovie(); ← We will code this method to launch
                        AddEditMovie activity to add movie
            return true;
        default:
            return super.onOptionsItemSelected(item);
    }
}
```

Launching Show Movie for Result, with request code for ADD

Movies.java

```
public static final int EDIT_MOVIE_CODE=1;
public static final int ADD_MOVIE_CODE=2;
...
public void addMovie() {
    Intent intent = new Intent(this, AddEditMovie.class);
    startActivityForResult(intent, ADD_MOVIE_CODE);
}
...
```

AddEditMovie finishes

- `onActivityResult` method distinguishes between return from Show/Edit and Add

```
@Override
protected void onActivityResult(int requestCode,
                                int resultCode,
                                Intent intent) {
    ...
    if (requestCode == EDIT_MOVIE_CODE) {
        Movie movie = movies.get(index);
        movie.name = name;
        movie.year = year;
        movie.director = director;
    } else if (requestCode == ADD_MOVIE_CODE) {
        movies.add(new Movie(name, year, director));
    }

    // redo the adapter to reflect change
    listView.setAdapter(
        new ArrayAdapter<Movie>(this,
                                R.layout.movie, movies));
    ...
}
```

index of movie in movies list,
that was passed through to
AddEditMovie

Adapter has to be redone
since source content has changed

11/21/16

Part 5: Non-raw file I/O, Android Device Monitor

<https://developer.android.com/guide/topics/data/data-storage.html#filesInternal>

Movies Input List

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    ..
    // load movies from file, or if no file, from string array
    try {
        FileInputStream fis = openFileInput("movies.dat");
        BufferedReader br = new BufferedReader(
            new InputStreamReader(fis));
        String movieInfo=null; movies = new ArrayList<Movie>();
        while ((movieInfo = br.readLine()) != null) {
            String[] tokens = movieInfo.split("\\|");
            if (tokens.length == 3) {
                movies.add(new Movie(tokens[0],tokens[1], tokens[2]));
            } else { movies.add(new Movie(tokens[0], tokens[1])); }
        }
    } catch (IOException e) {
        // load from bootstrap list in string resources
        String[] moviesList = getResources().getStringArray(...);
        movies = new ArrayList<Movie>(moviesList.length);
        for (int i=0; i < moviesList.length; i++) {
            String[] tokens = moviesList[i].split("\\|");
            movies.add(new Movie(tokens[0],tokens[1]));
        }
    }
}
```

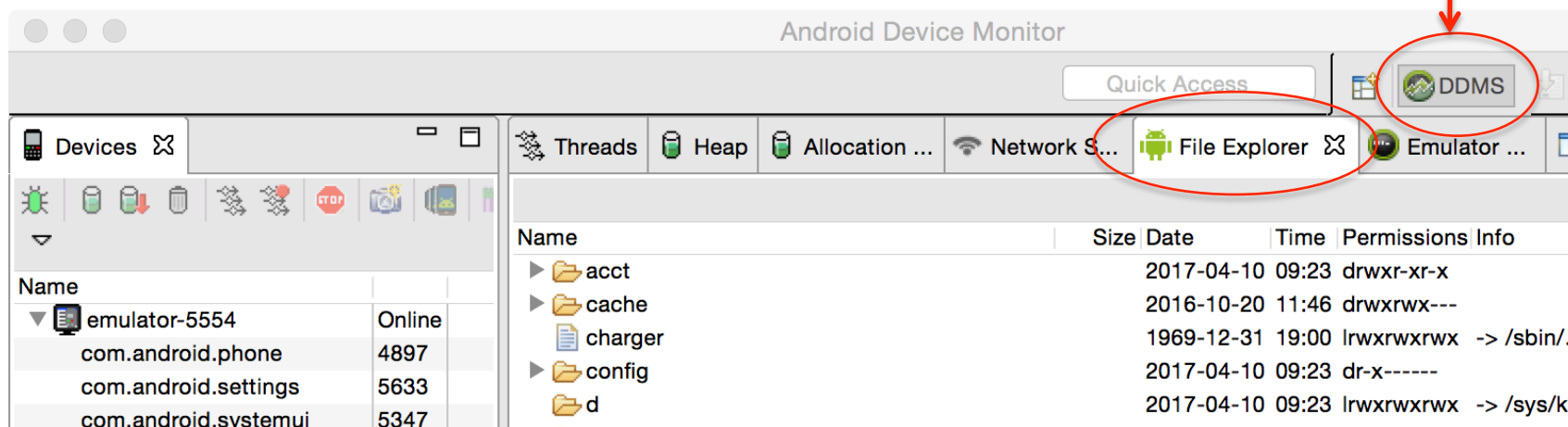
Setting up a file: Android Device Monitor

See <https://developer.android.com/studio/profile/monitor.html>

In Android Studio, do Tools -> Android -> Android Device Monitor

With the emulator running, you should see this:

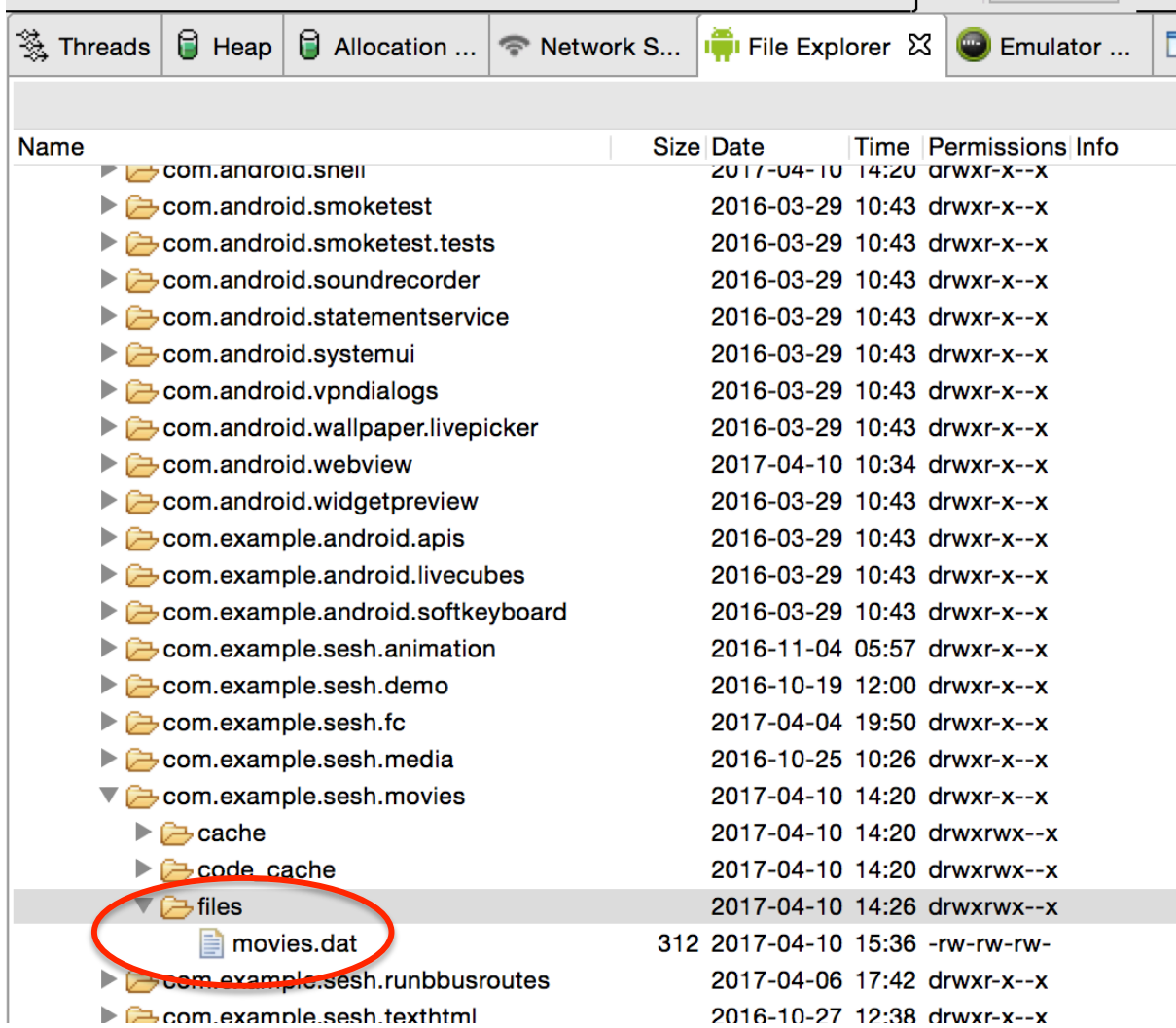
Dalvik Debug Monitor Server



Name	Size	Date	Time	Permissions	Info
▶ folder acct		2017-04-10	09:23	drwxr-xr-x	
▶ folder cache		2016-10-20	11:46	drwxrwx---	
file charger		1969-12-31	19:00	lrwxrwxrwx	-> /sbin/.
▶ folder config		2017-04-10	09:23	dr-x-----	
folder d		2017-04-10	09:23	lrwxrwxrwx	-> /sys/k

Working with Device Filesystem

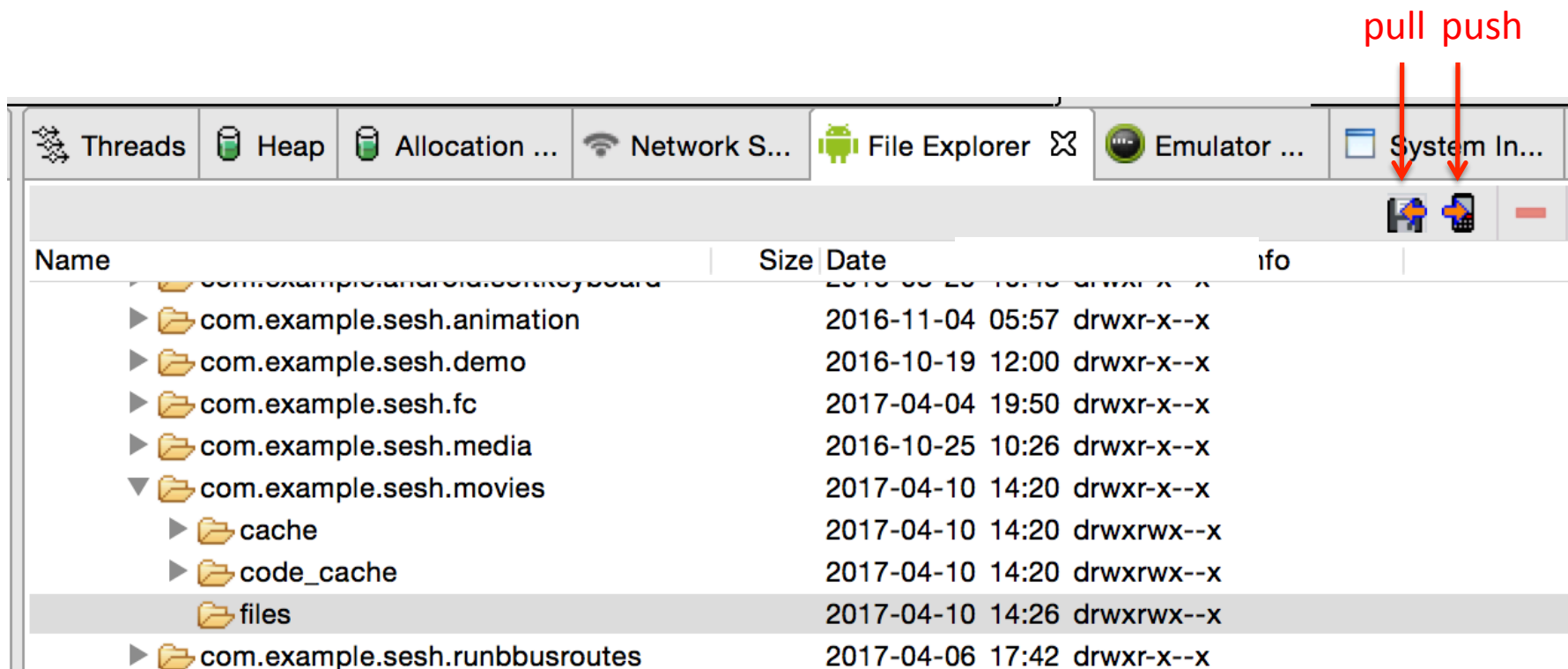
In File Explorer, you should see the file under [data->data->com.example.sesh.movies->files](#):



Name	Size	Date	Time	Permissions	Info
▶ com.android.sHELL		2017-04-10	14:20	drwxr-x--x	
▶ com.android.smoketest		2016-03-29	10:43	drwxr-x--x	
▶ com.android.smoketest.tests		2016-03-29	10:43	drwxr-x--x	
▶ com.android.soundrecorder		2016-03-29	10:43	drwxr-x--x	
▶ com.android.statementservice		2016-03-29	10:43	drwxr-x--x	
▶ com.android.systemui		2016-03-29	10:43	drwxr-x--x	
▶ com.android.vpndialogs		2016-03-29	10:43	drwxr-x--x	
▶ com.android.wallpaper.livepicker		2016-03-29	10:43	drwxr-x--x	
▶ com.android.webview		2017-04-10	10:34	drwxr-x--x	
▶ com.android.widgetpreview		2016-03-29	10:43	drwxr-x--x	
▶ com.example.android.apis		2016-03-29	10:43	drwxr-x--x	
▶ com.example.android.livecubes		2016-03-29	10:43	drwxr-x--x	
▶ com.example.android.softkeyboard		2016-03-29	10:43	drwxr-x--x	
▶ com.example.sesh.animation		2016-11-04	05:57	drwxr-x--x	
▶ com.example.sesh.demo		2016-10-19	12:00	drwxr-x--x	
▶ com.example.sesh.fc		2017-04-04	19:50	drwxr-x--x	
▶ com.example.sesh.media		2016-10-25	10:26	drwxr-x--x	
▼ com.example.sesh.movies		2017-04-10	14:20	drwxr-x--x	
▶ cache		2017-04-10	14:20	drwxrwx--x	
▶ code_cache		2017-04-10	14:20	drwxrwx--x	
▼ files		2017-04-10	14:26	drwxrwx--x	
movies.dat	312	2017-04-10	15:36	-rw-rw-rw-	
▶ com.example.sesh.runbbusroutes		2017-04-06	17:42	drwxr-x--x	
▶ com.example.sesh.txthtml		2016-10-27	12:38	drwxr-x--x	

Working with Device Filesystem

You can use the “pull” and “push” buttons to pull a file from device to local filesystem, or push a file from local filesystem to device



The screenshot shows the Android Studio interface with the File Explorer tab selected. The toolbar at the top includes buttons for Threads, Heap, Allocation, Network S..., File Explorer, Emulator, and System In... The File Explorer tab shows a list of files and folders for the device. The 'pull' and 'push' buttons are highlighted with red arrows and labels.

Name	Size	Date	Permissions
com.example.sesh.animation		2016-11-04 05:57	drwxr-x--x
com.example.sesh.demo		2016-10-19 12:00	drwxr-x--x
com.example.sesh.fc		2017-04-04 19:50	drwxr-x--x
com.example.sesh.media		2016-10-25 10:26	drwxr-x--x
com.example.sesh.movies		2017-04-10 14:20	drwxr-x--x
cache		2017-04-10 14:20	drwxrwx--x
code_cache		2017-04-10 14:20	drwxrwx--x
files		2017-04-10 14:26	drwxrwx--x
com.example.sesh.runbbusroutes		2017-04-06 17:42	drwxr-x--x