



JÖNKÖPING UNIVERSITY

School of Engineering

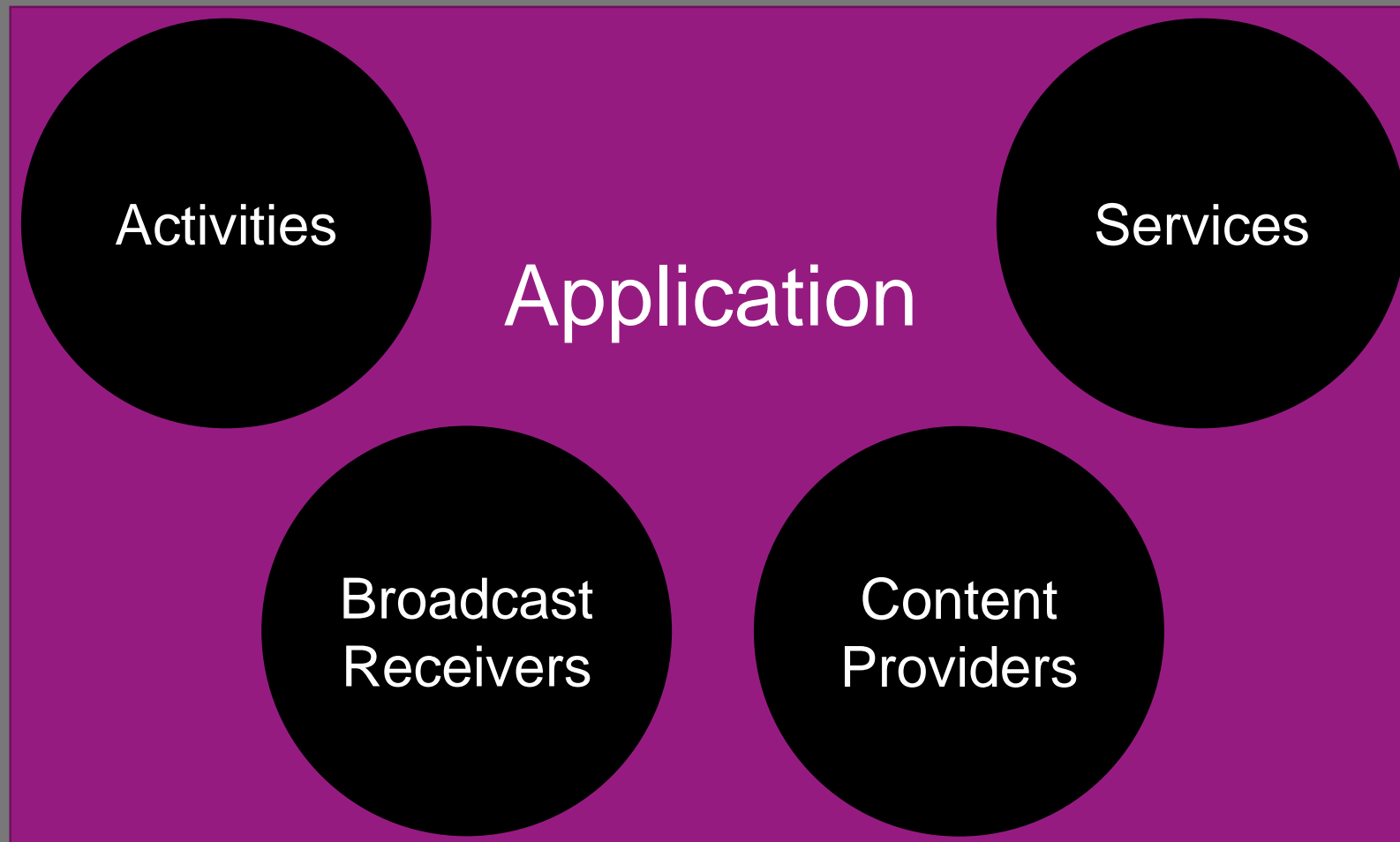
ANDROID CONTENT PROVIDERS

Peter Larsson-Green

Jönköping University

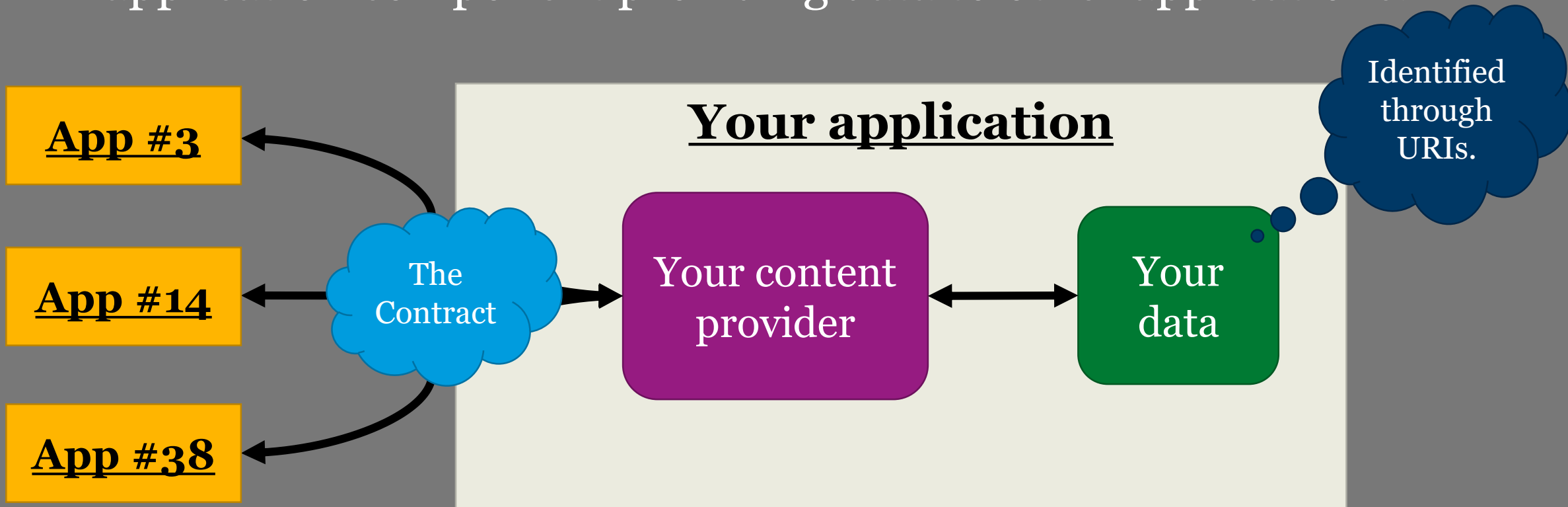
Spring 2020

FUNDAMENTAL APP COMPONENTS



WHAT'S A CONTENT PROVIDER?

An application component providing data to other applications.



WHAT'S A CONTENT PROVIDER?

An application component providing data to other applications.

- In theory, the data can be stored in any way.
- In practice, it is easy to use data from SQLite.

HOW DO I USE A CONTENT PROVIDER?

```
ContentResolver contentResolver = aContext.getContentResolver();
```

```
contentResolver.query(theUri, ...);
```

```
contentResolver.insert(theUri, ...);
```

```
contentResolver.update(theUri, ...);
```

```
contentResolver.delete(theUri, ...);
```

THE URI FOR CONTENT PROVIDERS

Identifies data in providers.

```
content://com.android.contacts/contacts/52
```

Scheme

Authority

Directory

Id

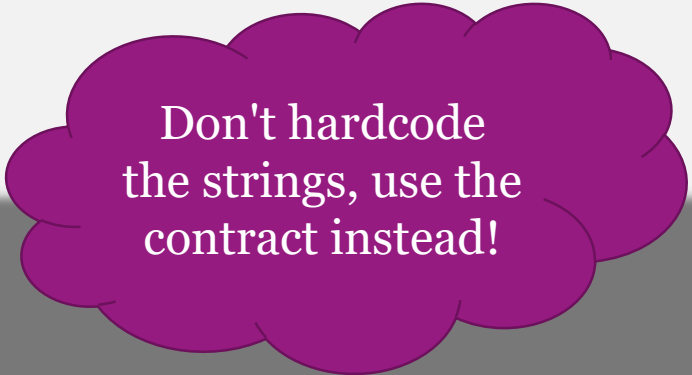
Useful methods:

```
Uri uri = Uri.parse("content://authority/collection");  
Uri uri2 = ContentUris.withAppendedId(uri, 37);  
long id = ContentUris.parseId(uri2);
```

READING DATA

```
contentResolver.query(  
    theUri,  
    theProjection,  
    theSelection,  
    theSelectionArgs,  
    sortOrder  
);
```

```
contentResolver.query(  
    Uri.parse("content://com.android.contacts/contacts"),  
    new String[]{ "display_name" },  
    "display_name = ?",  
    new String[]{ "Edsger W. Dijkstra" },  
    "display_name DESC"  
);
```



Don't hardcode
the strings, use the
contract instead!

```
<uses-permission android:name="android.permission.READ_CONTACTS"/>
```


READING DATA

```
Cursor cursor = contentResolver.query(...);  
  
int count = cursor.getCount();  
  
while (cursor.moveToNext()) {  
    String aString = cursor.getString(0);  
    int aNumber = cursor.getInt(1);  
}  
  
cursor.close();
```

INSERTING DATA

```
ContentValues values = new ContentValues();  
values.put("theColumn", theValue);
```

```
Uri uri = contentResolver.insert(  
    theUri,  
    values  
);
```

UPDATING DATA

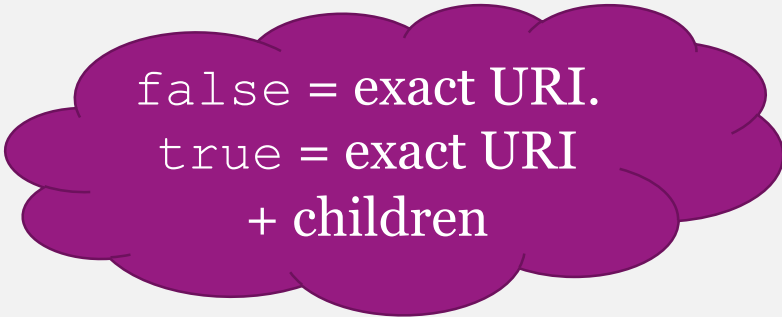
```
ContentValues values = new ContentValues();  
values.put("theColumn", theValue);  
  
int numberOfRows = contentResolver.update(  
    theUri,  
    values,  
    selection,  
    selectionArgs  
);
```

DELETING DATA

```
int numberOfAffectedRows = contentResolver.delete(  
    theUri,  
    selection,  
    selectionArgs  
);
```

LISTENING FOR DATA CHANGES

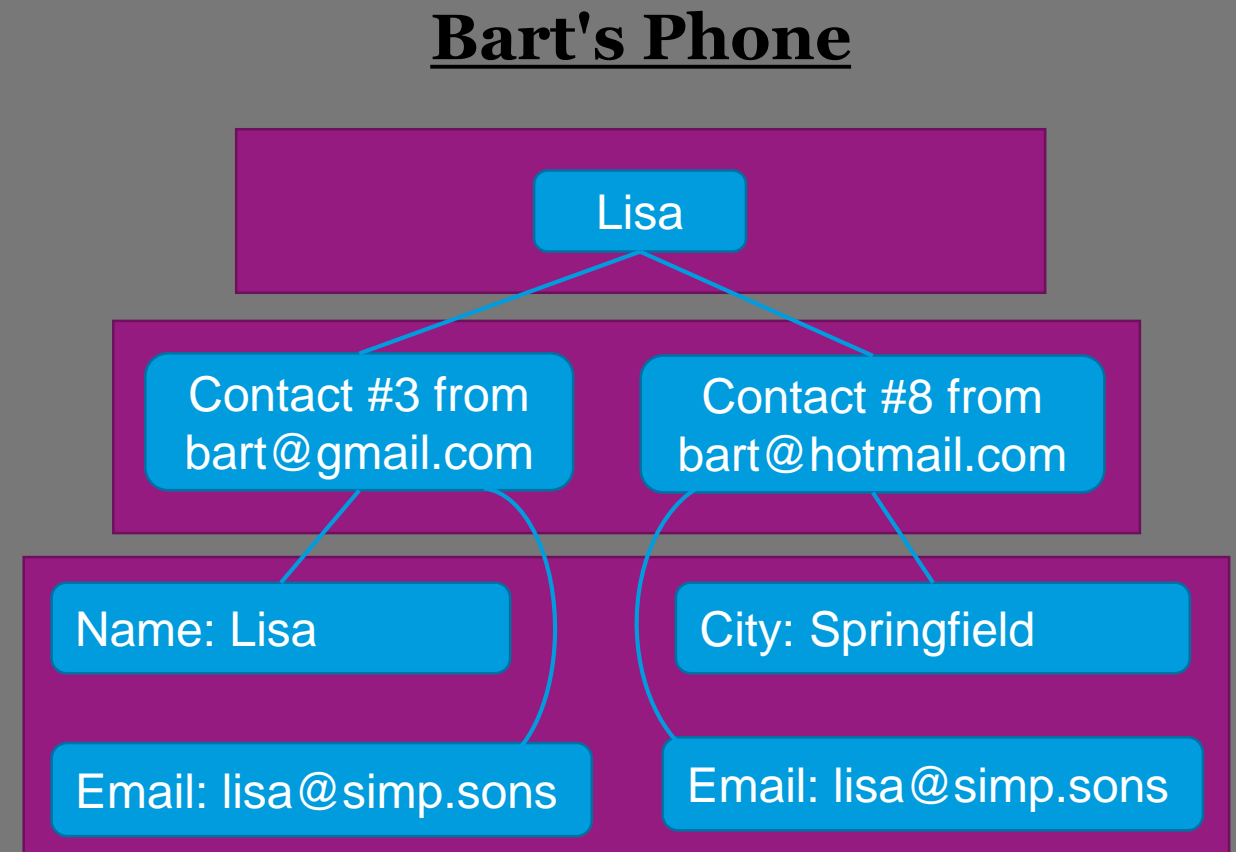
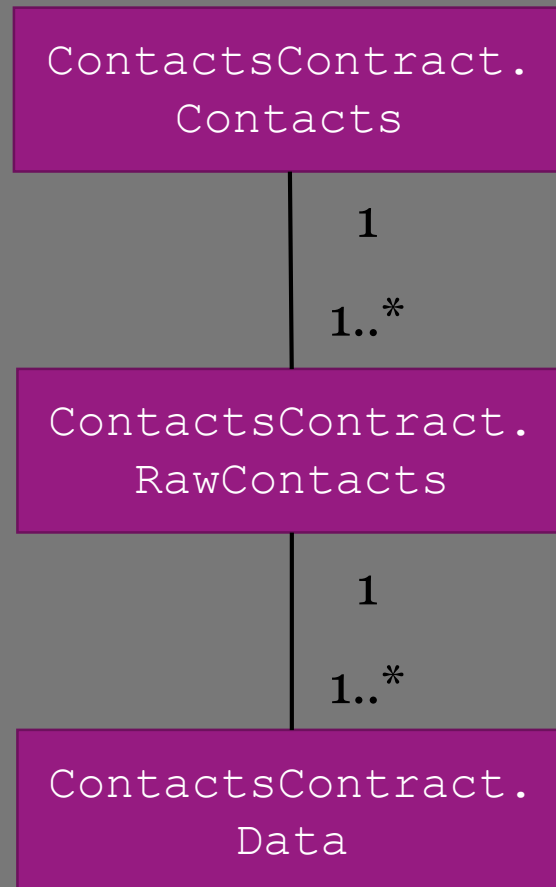
```
ContentObserver yourContentObserver = new ContentObserver() {  
    public ContentObserver() { super(new Handler()); }  
    public void onChange(boolean selfChange) { } /* API L <= 15 */  
    public void onChange(boolean selfChange, Uri uri) { } /* 16 <= API L */  
};  
contentResolver.registerContentObserver(  
    theUri,  
    false, • • •  
    yourContentObserver  
);
```



false = exact URI.
true = exact URI
+ children

```
contentResolver.unregisterContentObserver(yourContentObserver);
```

HOW CONTACTS ARE ORGANIZED



CONTACT PROVIDER'S CONTRACT

ContactsContract.Contacts.CONTENT_URI

ContactsContract.Contacts._ID,

ContactsContract.Contacts.DISPLAY_NAME

ContactsContract.CommonDataKinds.Phone.CONTENT_URI

ContactsContract.CommonDataKinds.Phone.NUMBER

ContactsContract.CommonDataKinds.Phone.CONTACT_ID

ContactsContract.CommonDataKinds.Email.CONTENT_URI

ContactsContract.CommonDataKinds.Email.ADDRESS

ContactsContract.CommonDataKinds.Email.CONTACT_ID

CREATING A CONTENT PROVIDER

```
<manifest package="the.package">
  <application ...>
    <provider
      android:name=".MyContentProvider"
      android:authorities="the.package.MyContentProvider"
      android:exported="true"
      android:readPermission="a.permission"
      android:writePermission="a.permission"
    />
  </application>
</manifest>
```


CREATING A CONTENT PROVIDER

```
public class MyContentProvider extends ContentProvider{  
    @Override  
    public boolean onCreate() {  
        return true;  
    }  
}
```



Did everything
go well?

CREATING A CONTENT PROVIDER

```
public class MyContentProvider extends ContentProvider{  
    public Cursor query(Uri uri, String[] projection, String selection,  
                        String[] selectionArgs, String sortOrder) { }  
  
    public Uri insert(Uri uri, ContentValues values) { }  
  
    public int delete(Uri uri, String selection, String[] selectionArgs) { }  
  
    public int update(Uri uri, ContentValues values, String selection,  
                     String[] selectionArgs) { }  
}
```

These methods must
be thread safe!

CREATING A CONTENT PROVIDER

```
public class MyContentProvider extends ContentProvider{  
    @Override  
    public String getType(Uri uri){  
        if(/* uri points to collection */){  
            return "vnd.android.cursor.dir/vnd.package.name";  
        }else{  
            return "vnd.android.cursor.item/vnd.package.name";  
        }  
    }  
}
```

ContentResolver.
CURSOR_DIR_BASE_TYPE

ContentResolver.
CURSOR_ITEM_BASE_TYPE

THE URI MATCHER

Zero or
more digits.

```
UriMatcher matcher = new UriMatcher(UriMatcher.NO_MATCH);  
matcher.addURI("the.authority", "the/path", 1);  
matcher.addURI("the.authority", "the/path-2", 2);  
matcher.addURI("the.authority", "the/path/#", 3);  
matcher.addURI("the.authority", "the/path-2/*", 4);
```

Zero or
more
characters.

```
Uri uri = Uri.parse("content://the.authority/the/path-2");  
int two = matcher.match(uri);
```


PATTERN FOR NOTIFYING CHANGES

Use content providers to notify changes.

- Need to properly implement `query`, `insert`, `update` & `delete`.
- To work properly, data may only be changed through these methods on the content provider.
 - In fragments/activities, work with the data through the content provider.

NOTIFYING CHANGES

```
public class MyContentProvider extends ContentProvider{  
    public Uri insert(Uri uri, ContentValues values){  
        // Do the insertion...  
        getContext().getContentResolver().notifyChange(  
            theUri,  
            theContentObserver.  
        );  
    }  
}
```



In many
cases null.

PROVIDING FILES

Content providers can also provide read and write streams to files.

```
ContentResolver contentResolver = aContext.getContentResolver();
```

```
InputStream is = contentResolver.openInputStream(theUri);
```

```
OutputStream os = contentResolver.openOutputStream(theUri, "w");
```

In your content provider, override:

```
openFile(Uri uri, String mode)
```

ADDING A FILE PROVIDER

```
<manifest package="the.package">
  <application ...>
    <provider
      android:name="androidx.core.content.FileProvider"
      android:authorities="se.ju.larpet.fileprovider"
      android:exported="false"
      android:grantUriPermissions="true">
      <meta-data
        android:name="android.support.FILE_PROVIDER_PATHS"
        android:resource="@xml/file_provider_paths"></meta-data>
    </provider>
  </application>
</manifest>
```


ADDING A FILE PROVIDER

```
<manifest package="the.package">
  <application ...>
    <provider
      android:name="android.support.v4.content.FileProvider"
      android:authorities="the.package.files"
      android:exported="true"
      android:grantUriPermissions="true">
      <paths>
        <files-path name="pics" path="cars/" />
      </paths>
      <meta-data
        android:name="android.support.FILE_PROVIDER_PATHS"
        android:resource="@xml/file_provider_paths"></meta-data>
    </provider>
  </application>
</manifest>
```

Part of URI
other apps
see.

Actual sub
directory.

EXAMPLE: TAKING PICTURE

```
Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);  
File folder = aContext.getFilesDir();  
File file = new File(folder, "cars/my-file.jpeg");  
Uri fileUri = FileProvider.getUriForFile(  
    aContext,  
    "se.ju.larpet.fileprovider",  
    file  
);  
intent.putExtra(MediaStore.EXTRA_OUTPUT, fileUri);  
intent.flags = Intent.FLAG_GRANT_WRITE_URI_PERMISSION  
startActivityForResult(intent, 1234);
```