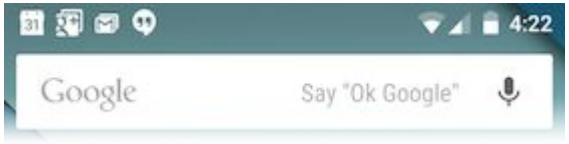
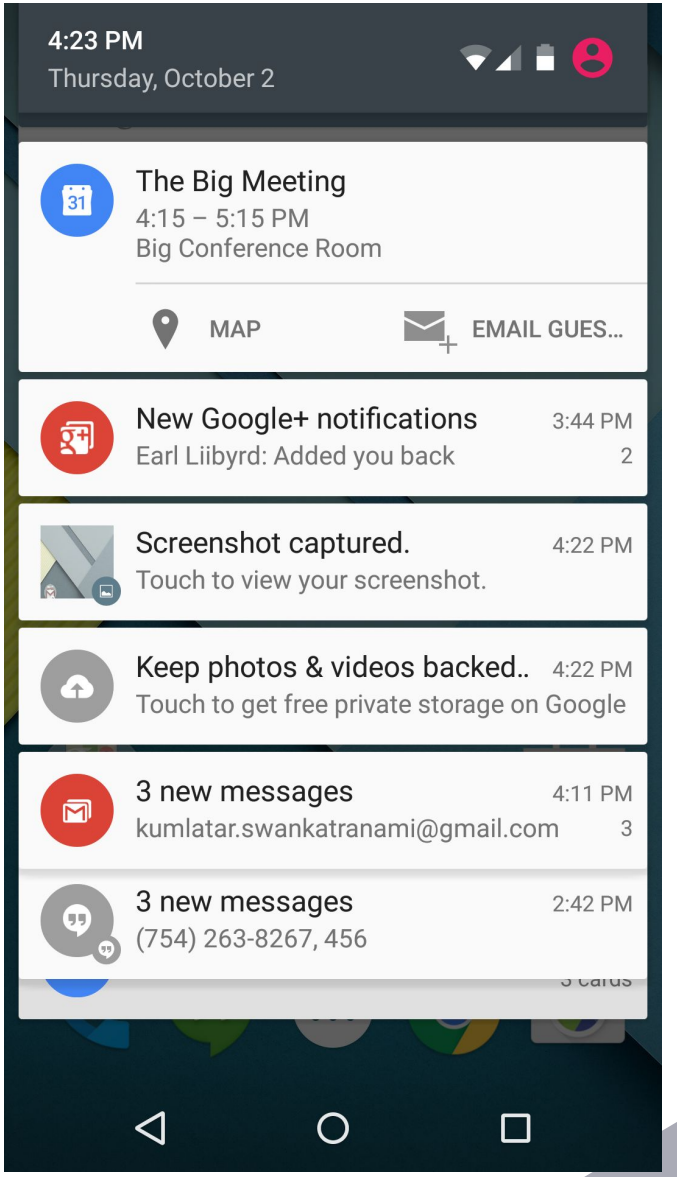
Android - Notifications, ListViews, Data persistence

Lukas Prokop

Notifications



- Notify user, when not use your app
- Mandatory
  - Small icon
  - Content title
  - Content text
  - Notification channel



avast

Notification

- Use `NotificationCompat.Builder`
  - Handles compatibility
- `NotificationManager.notify(int id, Notification)`
- Possible to set pending intent for actions
- Priority affects position in drawer
- Developer responsibility to handle navigation when user opens application from notification

Notification

- Updating notification
  - Call `NotificationManager.notify()` with the same id
- Remove notification
  - By swiping out setAutoCancel() on builder object
  - `cancel(int id)` delete notification with given id
  - `cancelAll()`
- Progress in Notification
  - Calling `setProgress(int max, int progress, boolean indeterminate)`
  - Update as regular notification
- Metadata
  - Category, priority, person

Heads-up notifications

- Heads-up notifications
  - Small floating window
  - Shows action buttons to handle action without leaving app
  - Only for high priority notifications
    - If the user's activity is in full screen mode (app uses fullScreenIntent)
    - Notification has high priority and uses ringtones or vibrations

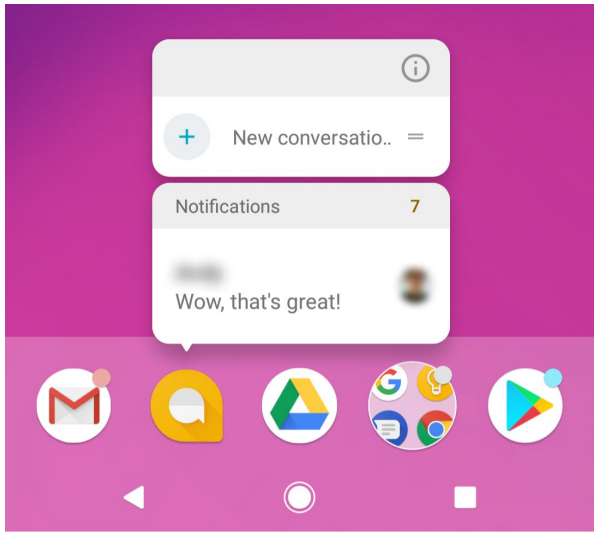
avast

Lock screen notifications

- Possibility to set which informations when device is locked
  - VISIBILITY_PUBLIC - Shows full content
  - VISIBILITY_SECRET - Do not show at all
  - VISIBILITY_PRIVATE - Show icon and title, hide content

Changes in Oreo

- Notification channels and groups
  - Allow user to manage notifications in channel
  - Behavior can’t be changed programmatically when channel already exists
- Notification badges (dots)
  - Dots on launcher app icons
  - Shows notification content on long click



avast

Exercises

1. Show notification when user data are downloaded
2. Show activity with list of repositories

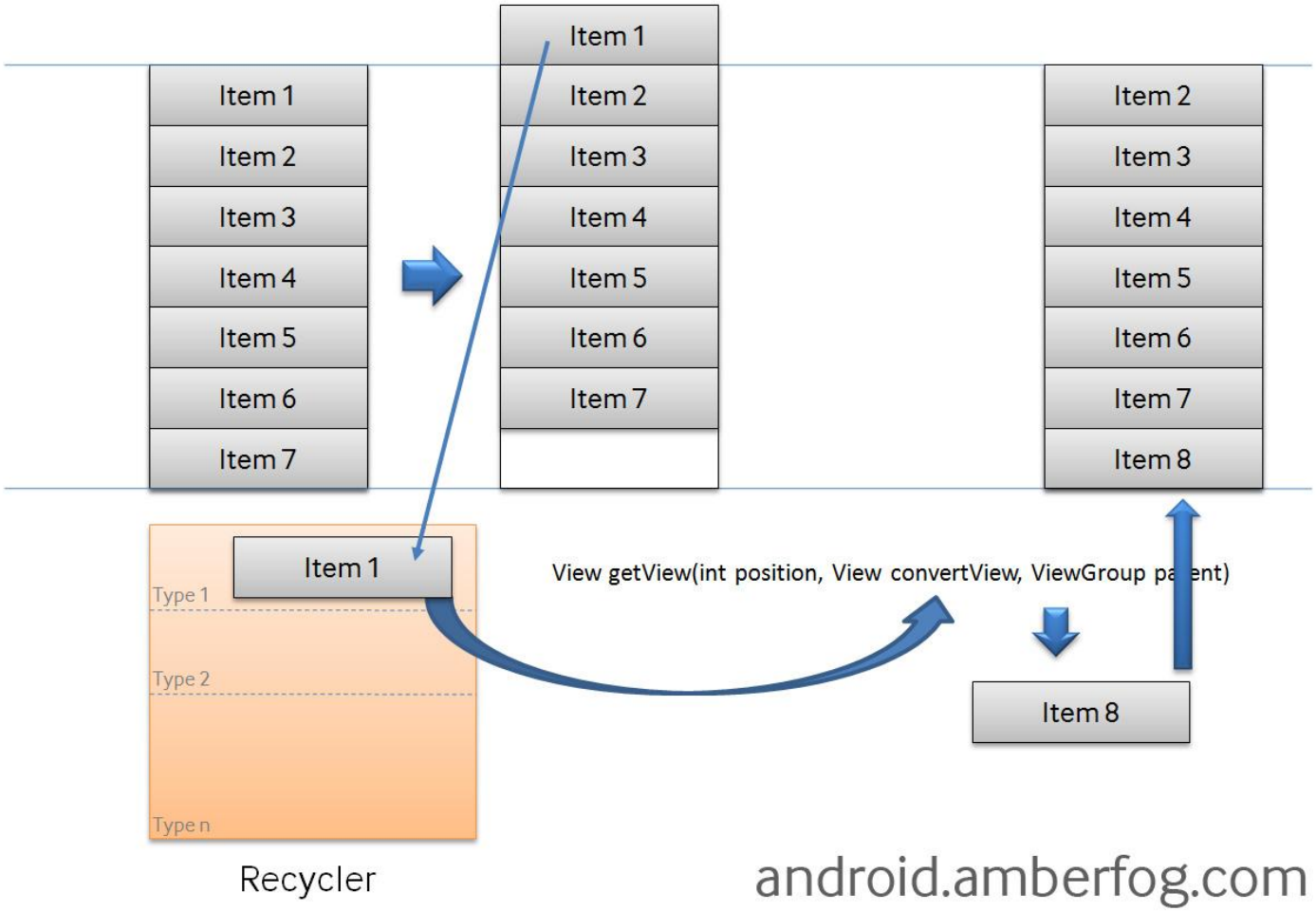
AdapterViews

- Views hold multiple items
- Horizontal scrolling
  - ListView
  - GridView
  - Spinner

Adapter

- Bridge between data and view
- Responsible for creating view for every item
- For inserting items into ListView, Spinner
- BaseAdapter
  - Common base implementation of adapter
  - `int getCount()`
  - `Object getItem(int position)`
  - `getItemId(int position)`
  - `View getView(int position, View convertView, ViewGroup parent)`
- Subclasses
  - ArrayAdapter<T>
  - CursorAdapter, SimpleCursorAdapter

View recycling



ViewHolder pattern

- Remember views
- findViewById is expensive operation
  - Traversing view for complex item
  - Impact on scroll smoothness

RecyclerView

- Part of support library v7
- Use holder pattern, simplify recycling
- Multiple Layout managers

Exercise

3. Use better layout for list of repositories and use ViewHolder pattern
4. Use Recycler adapter for users

Persisting data

- Files
- SharedPreferences
- Database
- ContentProvider

Persisting data - files

- Standard Java API for file operations
- Internal Storage
  - Always available
  - For private data
  - Removed with application uninstall
  - Cache
- External Storage
  - External storage != SD Card
  - Not always available
  - World readable
  - Uninstall remove files in `Context.getExternalFilesDir()`
  - Lot of API changes between android versions

Internal storage

- `Context.getFilesDir()`
  - File representing internal directory for your app
- `Context.openFileOutput(String, int)`
  - Filename - name of file
  - Mode - specify access to file
    - `MODE_PRIVATE` - accessible by apps with same UID
    - `MODE_APPEND` - append data instead of erasing file
    - `MODE_WORLD_READABLE` - Deprecated API 17, SecurityException API 24
    - `MODE_WORLD_WRITEABLE` - Deprecated API 17, SecurityException API 24
- `Context.openFileInput(String)`
  - Filename - name of file

Internal storage - cache

- `Context.getCacheDir()`
  - File representing internal directory for app temporary files
  - System can delete these files, when is running low on storage
  - Delete these files when are not longer needed
  - Presence of these files should not affect your application
    - It can just slow down app, need to download some resources

Internal storage - sharing data

- Data can be shared via `FileProvider`
  - Allows to specify shared directories
  - Implicit intent to pick specific files

External storage

- Requires permissions
  - `android.permission.WRITE_EXTERNAL_STORAGE`
  - `android.permission.READ_EXTERNAL_STORAGE`
  - Since API 19 permissions are not needed for private files
- It's your responsibility to check if the external storage is available
- Public files
  - Available to the other apps and user
  - Downloaded files
- Private files
  - Files to be deleted with app uninstall
  - Accessible to other, but no value for them
  - Temp downloaded files, ringtones, ….

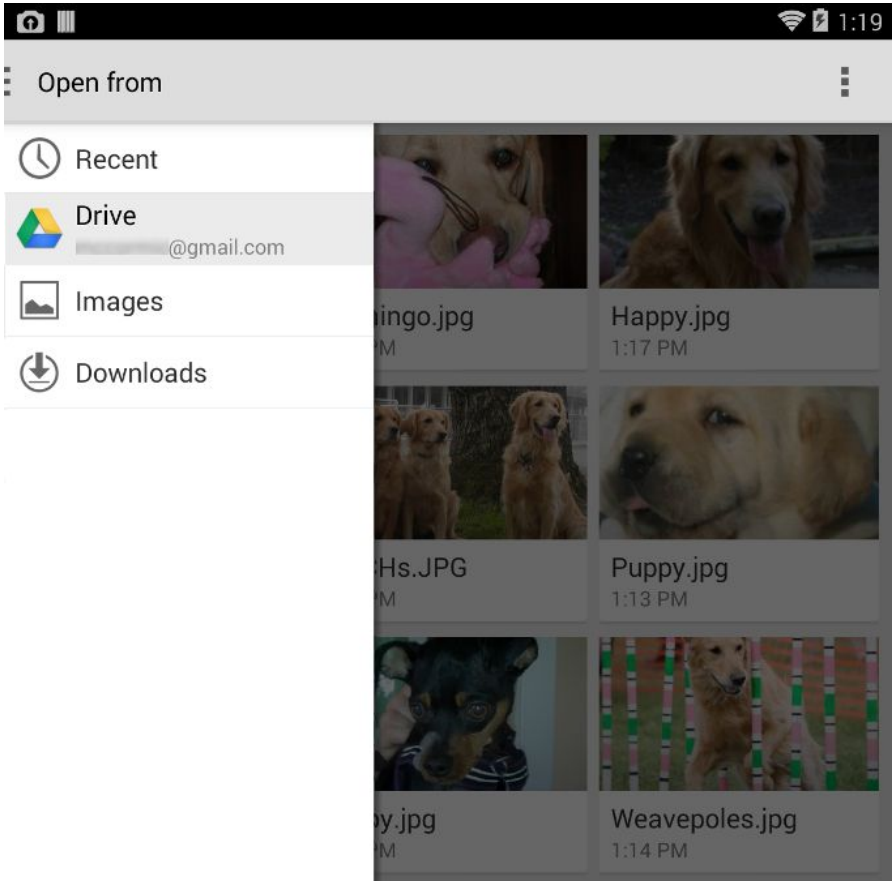
External storage

- `Environment.getExternalStoragePublicDirectory(String type)`
  - Type - type of files to access `Environment.DIRECTORY_*`
  - File representing top-level shared/external directory for files of particular type
  - Multi user devices - access only to current user
- `Environment.getExternalFilesDir(String type)`
  - Type - type of files to access `Environment.DIRECTORY_*`
  - `File` representing where app place internal files
  - Files are deleted after app uninstall
- `Environment.isExternalStorageEmulated()`
- `Environment.isExternalStorageRemovable()`
- `Environment.getExternalStorageState()`

External storage - sdcard

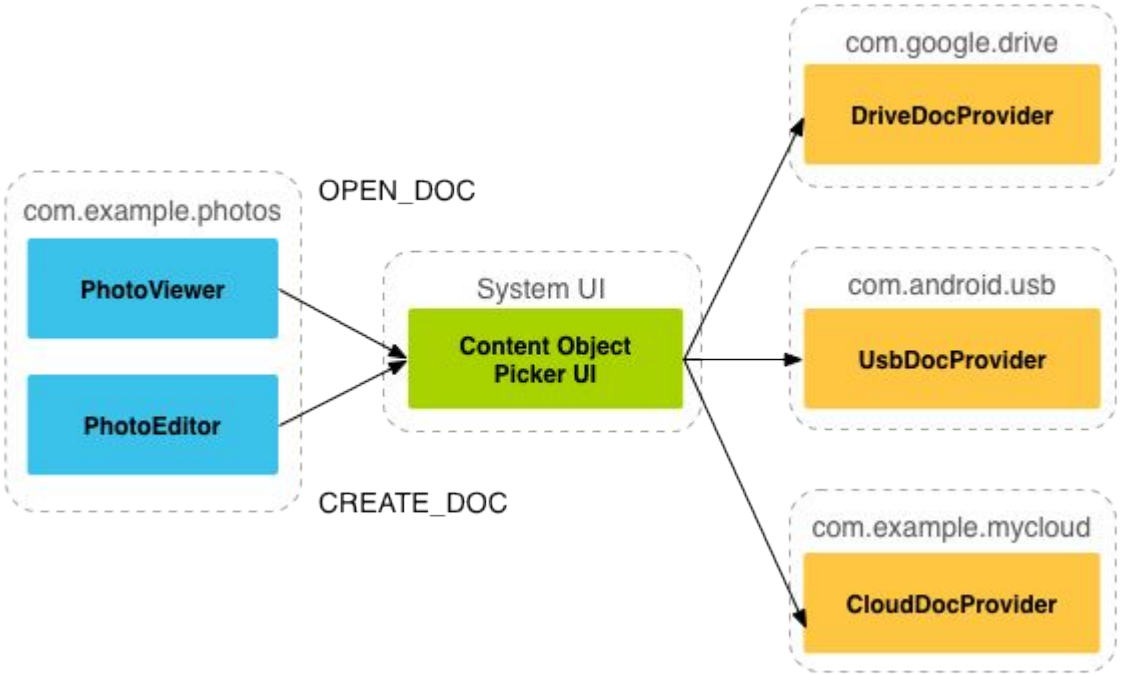
- < API-19 guess where the sdcard is mounted
- = API-19 not possible write shared data on sd card, when primary external storage is available
  - Or using storage access framework, but access is granted per file
- >=API-21 Storage access framework allows to grant access for directories

Storage access framework



- Let user pick a file
- Allows to plug-in custom service (cloud services like Dropbox, Google drive, …)
- Since API 19

Storage access framework



SharedPreferences

- Key value storage
- Backed by XML
- `Context.getSharedPreferences(String name, int mode)`
  - Name - name of file with preferences
  - Mode - operating mode
    - `MODE_PRIVATE` - only apps with same UID have access
    - `MODE_WORLD_READABLE` - API 17 Depracated, API 24 SecurityException
    - `MODE_WORLD_WRITEABLE` - API 17 Depracated, API 24 SecurityException
- `Activity.getPreferences(int mode)`
  - Preferences associated with activity
- `PreferenceManager.getDefaultSharedPreferences(Context)`
  - Default preferences used by Preference framework

SharedPreferences

```
SharedPreferences sharedPreferences = getSharedPreferences("preferences",
MODE_PRIVATE);
int intValue = sharedPreferences.getInt("int_key", 42);
String stringValue = sharedPreferences.getString("string_key", "Default
value");
```

```
SharedPreferences.Editor editor = sharedPreferences.edit();
editor.putString("string_key", "new value");
editor.commit();
editor.apply();
```

SharedPreferences

- Editor.commit()
  - Notifies about result
  - Synchronous operation, waits until changes are written to disk
- Editor.apply()
  - Async variant
  - Atomically stores values
- If multiple editors modifying preferences at she same time, last calling apply() wins
- Debugging - rooted device or stetho

Exercise

5. Count app launches
6. Use stetho for inspect shared preferences

Database - SQLite

- Full-featured SQL
- Single-file database
- Source code is just 1 file
- Small footprint
- ACID transactions
- Well documented
- Supports most of the SQL92 standard

SQLite on android

- Foreign keys disabled by default
- Internal storage
- Collation
  - BINARY - SQLite default
  - LOCALIZED - changes with system locale
  - UNICODE - Unicode collation algorithm
- Thread safe
- Create/upgrade on background thread
- Take care about opening/closing from different threads
- Use `BaseColumn._ID` for primary keys, some components rely on it
- Stetho tool for debugging

Database

- `SQLiteOpenHelper`
  - Database creation
  - Version management
  - Sqlite configuration
    - Enable write ahead log
    - Enable support for foreign keys
- `SQLiteDatabase`
  - Exposes methods to manage a SQLite databases
  - CRUD methods
  - Manage transactions

SQLiteOpenHelper

- `onCreate(SQLiteDatabase db)`
  - Called when the database is created for the first time
- `onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)`
  - Upgrade logic
- `getReadableDatabase/getWriteableDatabase`
  - creates/open database
  -
- `close()`
  - Close open database object

SQLiteDatabase

- `long insert(String table,`
  `String null,`
  `ContentValues values)`
  - Table - name of table
  - nullColumnHack - optional, allows to insert empty row
  - Values - inserted values
  - Returns ID of newly inserted row
- `long insertOrThrow`
- `long insertWithOnConflict`

SQLiteDatabase

- `Cursor query(boolean distinct,`
  `String table, String[] columns,`
  `String selection, String[] selectionArgs,`
  `String groupBy, String having,`
  `String orderBy, String limit)`
  - `Selection` - WHERE clausule, values replaced by ?
  - `selectionArgs` - values to replace ? in selection
- Multiple variants of query, with different possibilities
- `Cursor rawQuery(String sql,`
  `String[] selectionArgs)`
- Close returned cursors

SQLiteDatabase

- `int update(String table,`
  `ContentValues values,`
  `String whereClause,`
  `String[] whereArgs)`

SQLiteDatabase

- `int delete(String table,`
  `ContentValues values,`
  `String whereClause,`
  `String[] whereArgs)`

SQLiteDatabase

- Every CRUD operation is a transaction
- For inserting more rows in one time use transactions
- `beginTransaction()`
- `endTransaction()`
- `setTransactionSuccessful()`

ContentProvider

- Access to structured set of data
- Define data security
  - Via permissions
    - Global
    - Read/Write permissions
    - For single URI
- Connects data from one process to code running in another process
- ContentResolver for access data

ContentProvider

- Used by system aps
  - SMS
  - Contacts
  - Calendar
- Allows to share data between aps
- Data specified via Uri
- Allows to use CursorLoader

ContentProvider

- Can be backed up by different data sources
  - SQLite database
  - Network
  - Files
  - …

ContentProvider - implementation

1. Design data storage
2. Design content URIs
   a. `content://com.example.app.provider/table1`
   b. `content://com.example.app.provider/table2/dataset1`
   c. `content://com.example.app.provider/table3/#`
3. Define UriMatcher
   a. Translates Uris to number constant
4. Extend ContentProvider class
   a. query(), insert(), update(), delete()
   b. getType()
   c. onCreate() - fast operations, postpone db creation
5. Register provider in manifest

ContentResolver

- `context.getContentResolver()`
- CRUD operations similar params as SQLiteDatabase
- Specify data by URI

Libraries

- ButterKnife
  - Field and method binding for android views
  - http://jakewharton.github.io/butterknife/
- Retrofit
  - A type-safe HTTP client for Android and Java
  - http://square.github.io/retrofit/
- OkHttp
  - HTTP client for android and Java
  - http://square.github.io/okhttp/
- Stetho
  - Debug tool from facebook
  - http://facebook.github.io/stetho/

Libraries

- Dagger
  - Dependency injection library
  - https://google.github.io/dagger/


avast
Thanks for attention
prokop@avast.com