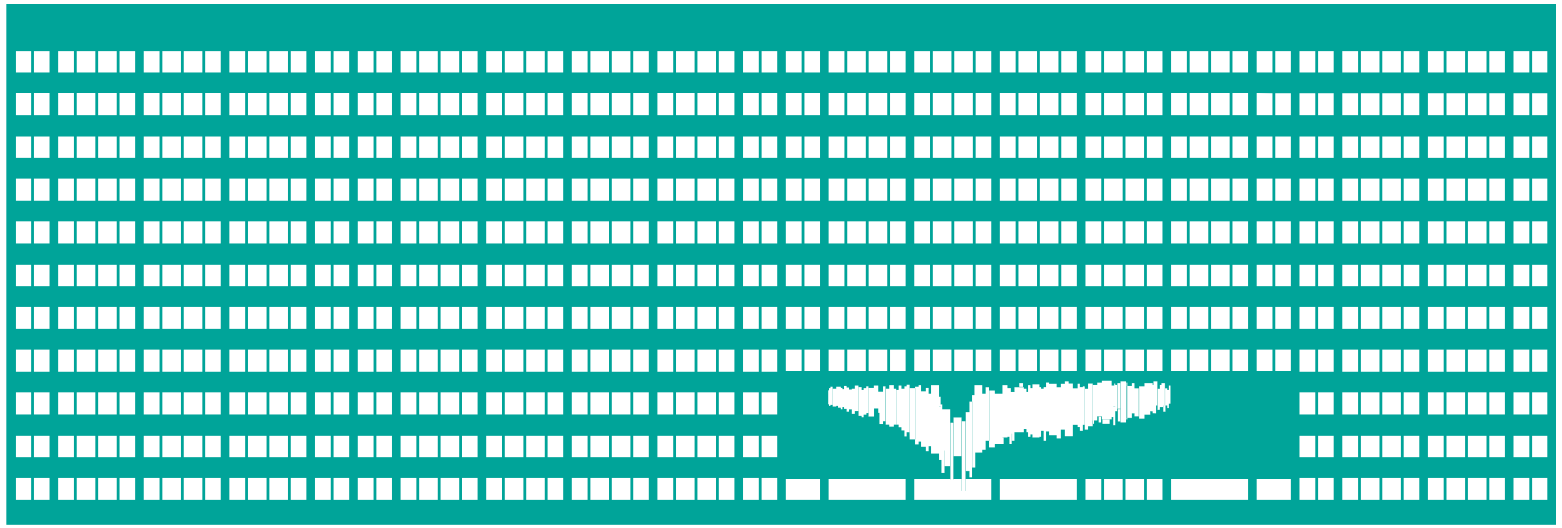
VŠB TECHNICKÁ UNIVERZITA OSTRAVA

VSB TECHNICAL UNIVERSITY OF OSTRAVA



www.vsb.cz

10 Useful/Interesting Android Libraries

Michal Krumnikl

Introduction

- Many Android developers have created useful libraries
- Libraries are available in a Maven repository.
- It is simple to add them to Android Studio project.
- Most of them have permissive **licenses** so that you can use them for free and include in free/commercial apps/products.
  - Some libraries must be downloaded as .JAR and added manually.

Maven and Gradle

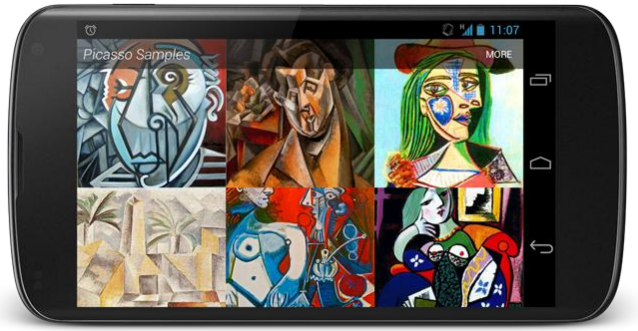
- **Maven Artifact** concept
  - Java “package” to be deployed
  - Different types of artifact (jar, war, plugin, …)
  - Dependent lifecycle
  - Phases for jar
    - process-resources, compile, process-test-resources, test-compile, test, package, install, deploy
  - Default behaviour
  - **Maven automatically updates and downloads the jar**
- **Gradle** builds upon the concepts of Apache Ant and Apache Maven
  - Introduces a **Groovy-based domain-specific language (DSL)**
  - Uses a directed acyclic graph (DAG)
  - Supports incremental builds

Dependency Resolution Management

- When your dependency is something other than a local library (**mavenLocal**) or file tree, Gradle looks for the files in whichever **online repositories** are specified in the settings.gradle file.
- By default, new Android Studio projects specify **Google's Maven repository**, and the **Maven central repository.**
  - Maven Central is a Maven Repository hosted by sonatype.org

```
dependencyResolutionManagement {
    repositoriesMode.set(RepositoriesMode.FAIL_ON_PROJECT_REPOS)
    repositories {
        google()
        mavenCentral()
        ivy(url = "https://repo.example.com/ivy")
        mavenLocal()
    }
}
```

Picasso



- **Hassle-free image loading in application**
  - Handling ImageView recycling and download cancelation in an adapter.
  - Complex image transformations with minimal memory use.
  - Automatic memory and disk caching.
  - Network donwloading / progress.

```
// https://github.com/square/picasso
implementation 'com.squareup.picasso:picasso:2.8'
```

```
String imageUri = "https://i.imgur.com/tGbaZCY.jpg";
ImageView ivBasicImage = (ImageView) findViewById(R.id.ivBasicImage);
Picasso.with(context).load(imageUri).into(ivBasicImage);

//resizing
Picasso.with(this).load(imageUri).resize(someWidth, 0).into(ivBasicImage);
```

COIL

- **Co**routine **I**mage **L**oader - image library for Android and Compose Multiplatform.
- Fetching images from network using OkHttp/Ktor2/Ktor3
- Caching, decoding, request management, memory management, …
- Decoders for **GIF, SVG, videos**

```
// https://coil-kt.github.io/coil/
implementation("io.coil-kt.coil3:coil-compose:3.0.3")
implementation("io.coil-kt.coil3:coil-network-okhttp:3.0.3")
```

```
imageView.load("https://example.com/image.jpg")
```

```
AsyncImage(
    model = "https://example.com/image.jpg",
    contentDescription = null,
)
```

Lottie

- Parses **Adobe After Effects animations** exported as Json with Bodymovin and renders them natively on mobile.

```
// https://github.com/airbnb/lottie-android
implementation 'com.airbnb.android:lottie:6.6.0'
```

```
<com.airbnb.lottie.LottieAnimationView
        android:id="@+id/animation_view"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:lottie_fileName="hello_world.json"

        // Loop indefinitely
        app:lottie_loop="true"
        // Start playing as soon as the anim is loaded
        app:lottie_autoPlay="true" />
```

```
LottieAnimationView animationView =
        findViewById(R.id.animation_view);
animationView.setAnimation(R.raw.your_animation);
animationView.setRepeatCount(2); // Play twice
animationView.setSpeed(1.5f); // Play speed 1.5x
animationView.playAnimation();
```

ZXing

- **Barcode scanner library** for Android, based on the **ZXing** decoder
  - https://github.com/zxing/zxing/
- Provides scanner activity and barcode generators

```
QRCodeEncoder qr = new QRCodeEncoder("Hello",
                                     null,
                                     Contents.Type.TEXT,
                                     BarcodeFormat.QR_CODE.toString(),
                                     smallerDimension);

Bitmap bitmap = qr.encodeAsBitmap();
```

Hello

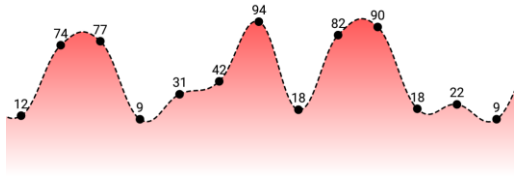
```
BarcodeFormat.UPC_A
BarcodeFormat.UPC_E
BarcodeFormat.EAN_13
BarcodeFormat.EAN_8
BarcodeFormat.RSS_14
BarcodeFormat.CODE_39
BarcodeFormat.CODE_93
BarcodeFormat.CODE_128
BarcodeFormat.ITF
BarcodeFormat.CODABAR
BarcodeFormat.QR_CODE
BarcodeFormat.DATA_MATRIX
BarcodeFormat.PDF_417
```

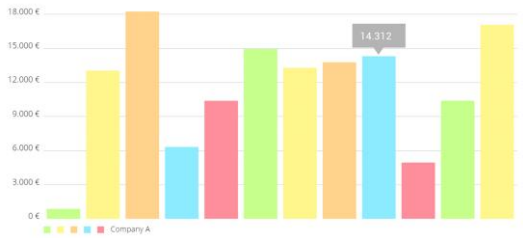
```
// https://github.com/journeyapps/zxing-android-embedded
implementation 'com.journeyapps:zxing-android-embedded:4.3.0'
```

MPAndroidChart

MPAndroidChart
created by Philipp Jahoda

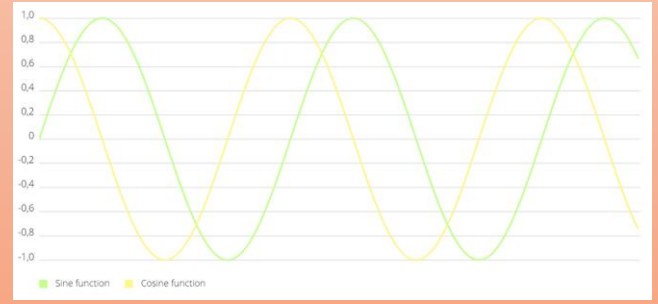
- **Various charts** - Line Chart, Bar Chart, Scatter Chart, Candle Stick Chart, Pie Chart, Bubble Chart or Radar Chart
- Provides i**nteraction and animations**
- Support **dynamic and realtime data**





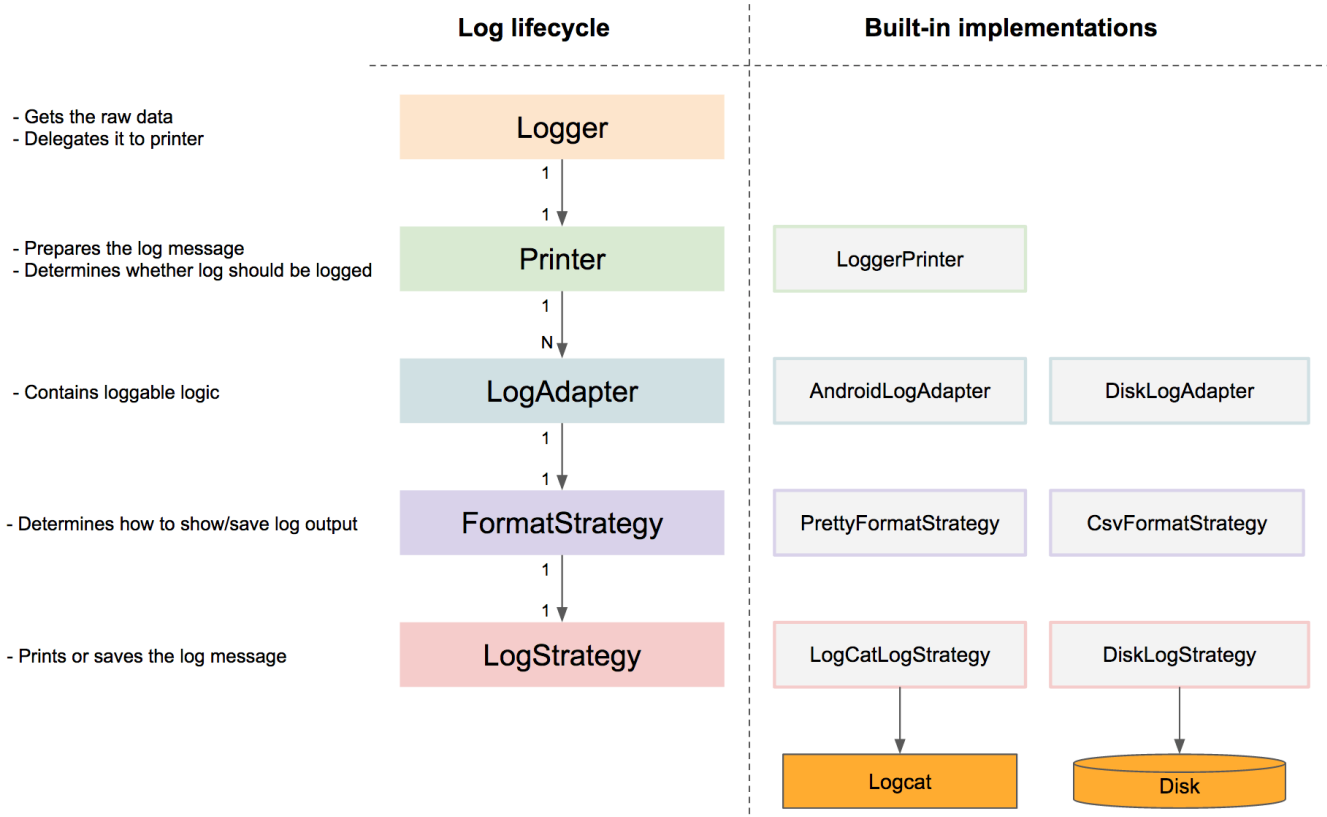

```
// https://weeklycoding.com/mpandroidchart/
implementation 'com.github.PhilJay:MPAndroidChart:v3.1.0'
```

```
LineChart chart = findViewById(R.id.chart);
chart.setData(data);
chart.invalidate();
```



Logger

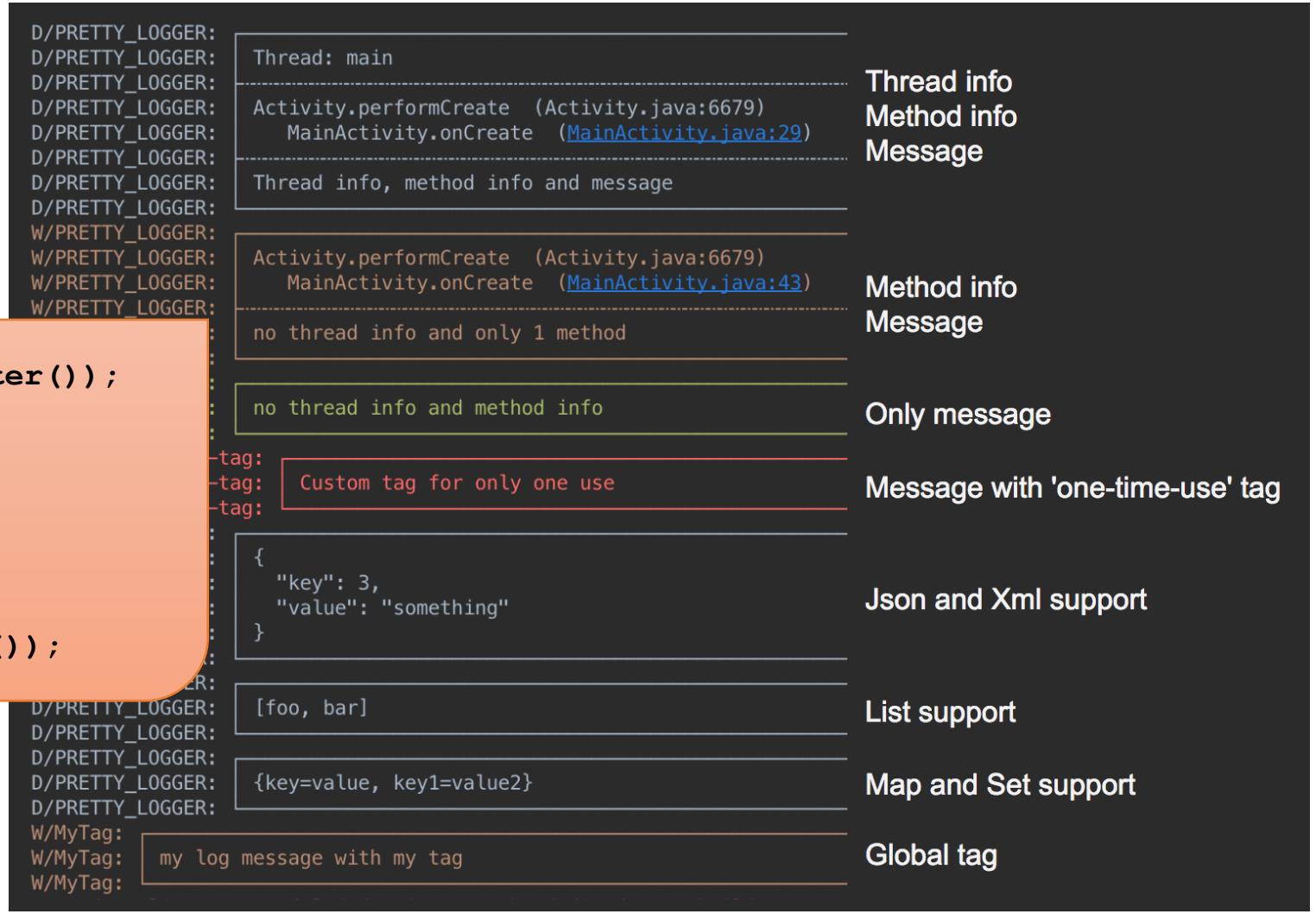
- **Simple, pretty logger for Android**
- Logging features
  - Thread, Class, Method information
  - **Pretty-print for Json and XML content**
  - Pretty-print for new line "\n"
  - Clean output
  - Save logs to file




```
// https://github.com/orhanobut/logger
implementation 'com.orhanobut:logger:2.2.0'
```

Logger

```
Logger.addLogAdapter(new AndroidLogAdapter());

Logger.d("hello");
Logger.e(exception, "message");
Logger.json(JSON_CONTENT);
Logger.xml(XML_CONTENT);

Logger.addLogAdapter(new DiskLogAdapter());
```



LeakCanary

- **Memory leak detection** library for Android
  - *For example, an Android Activity instance is no longer needed after its onDestroy() method is called, and storing a reference to that instance in a static field prevents it from being garbage collected.*
- Automatically detects leaks of the following objects:
  - destroyed Activity instances, destroyed Fragment and View instances
  - cleared ViewModel instances, destroyed Service instance
- Process:
  - **Detecting retained objects -> Dumping the heap -> Analyzing the heap -> Categorizing leaks**

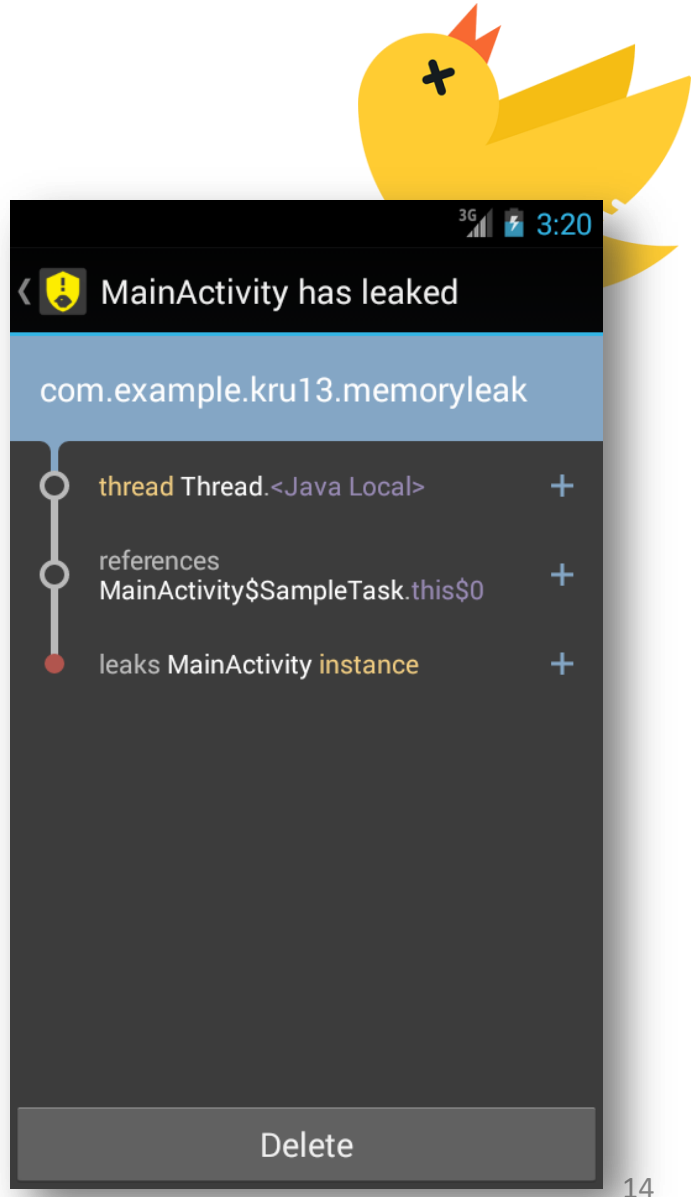
```
// https://square.github.io/leakcanary/
debugImplementation 'com.squareup.leakcanary:leakcanary-android:2.14'
```

LeakCanary

- **Common causes for memory leaks**
  - Adding a Fragment instance to the backstack without clearing that Fragment's view fields in Fragment.onDestroyView()
  - Storing an Activity instance as a Context field in an object that survives activity recreation due to configuration changes.
  - Registering a listener, broadcast receiver or RxJava subscription which references an object with lifecycle, and forgetting to unregister when the lifecycle reaches its end.
- **Result**
  - UI freezes and **Application Not Responding (ANR)** reports
  - **OutOfMemoryError (OOME)** crash.

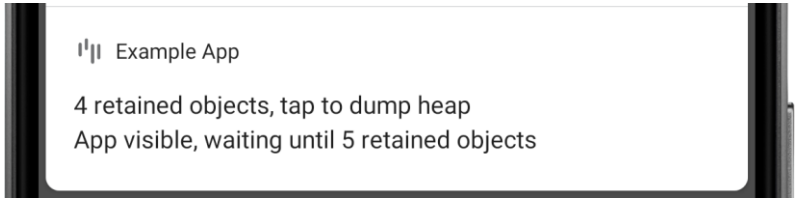
LeakCanary

```
public class MainActivity extends Activity {

  @Override protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main_activity);
     SampleTask();
  }

  private void SampleTask() {
  new AsyncTask<Void, Void, Void>() {
      @Override protected Void doInBackground(Void... params) {
        // Do some slow work in background
        SystemClock.sleep(20000);
        return null;
      }
    }.execute();
  }
}
```

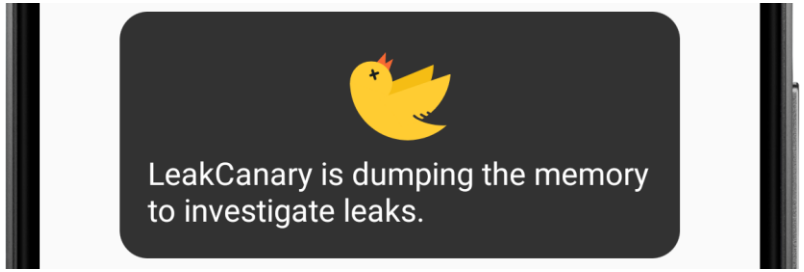


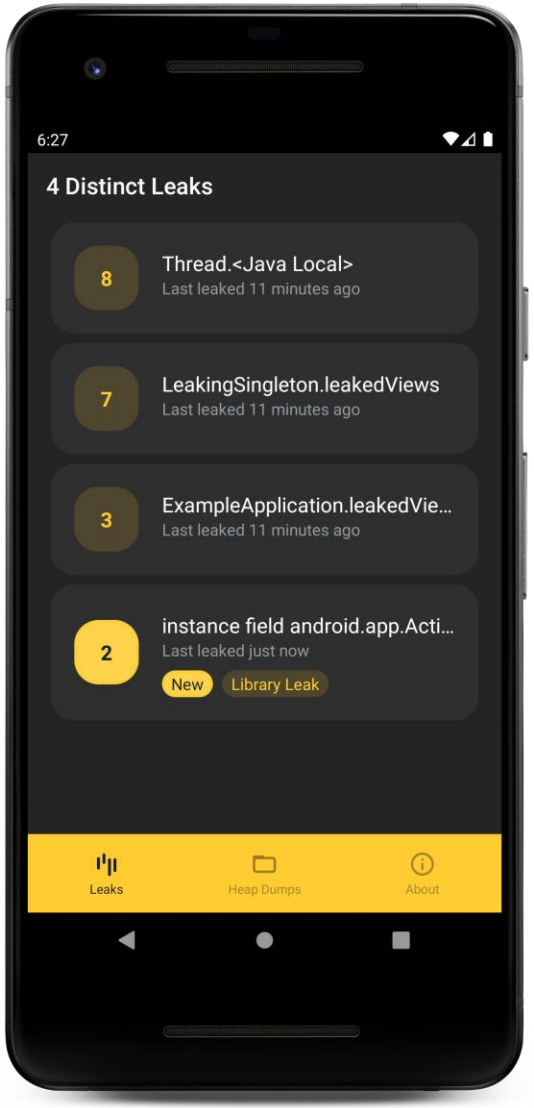
LeakCanary

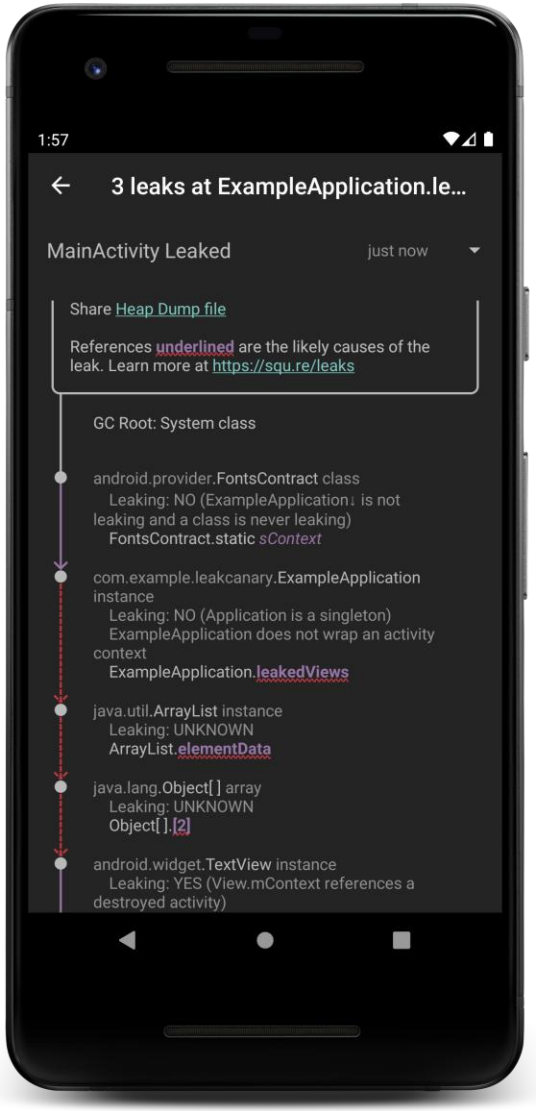
- LeakCanary waits for the count of retained objects to **reach a threshold**



- **Dumping and analyzing the heap**







Retrofit

- A **type-safe HTTP client** for Android and Java.
- Annotate to describe the HTTP request
- **Support converters**
  - Gson
  - Jackson
  - Moshi
  - Protobuf
  - Wire
  - Simple XML
  - JAXB

```
// https://square.github.io/retrofit/
implementation 'com.squareup.retrofit2:retrofit:2.11.0'
```

```
public interface GitHubService {
  @GET("users/{user}/repos")
  Call<List<Repo>> listRepos(@Path("user") String user);
}

Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("https://api.github.com/")
    .build();

GitHubService service = retrofit.create(GitHubService.class);
Call<List<Repo>> repos = service.listRepos("octocat");
```

GSON

- **Convert Java Objects into their JSON representation and vice versa.**
  - Other JVM languages such as Kotlin or Scala might work fine in many cases, but language-specific features such as Kotlin's non-null types or constructors with default arguments are not supported.
- Allow pre-existing unmodifiable objects to be converted to and from JSON.
- Support arbitrarily complex objects.
- Generate compact and readability JSON output

{ Gson }

```
// https://github.com/google/gson
implementation 'com.google.code.gson:gson:2.11.0'
```

GSON

- Serialization using **toJson()**

```
Gson gson = new Gson();
gson.toJson(1);             // ==> 1
gson.toJson("abcd");        // ==> "abcd"
gson.toJson(new Long(10)); // ==> 10
int[] values = { 1 };
gson.toJson(values);        // ==> [1]
//-----------------------------------------------------------------------//
class BagOfPrimitives {
  private int value1 = 1;
  private String value2 = "abc";
  private transient int value3 = 3;
  BagOfPrimitives() { // no-args constructor }
}

BagOfPrimitives obj = new BagOfPrimitives();
Gson gson = new Gson();
String json = gson.toJson(obj);       // ==> json is {"value1":1,"value2":"abc"}
```

GSON

- Deserialization using **fromJson()**

```
int one = gson.fromJson("1", int.class);
Integer one = gson.fromJson("1", Integer.class);
Long one = gson.fromJson("1", Long.class);
Boolean false = gson.fromJson("false", Boolean.class);
String str = gson.fromJson("\"abc\"", String.class);
String anotherStr = gson.fromJson("[\"abc\"]", String.class);
//-----------------------------------------------------------------------//
class BagOfPrimitives {
  private int value1 = 1;
  private String value2 = "abc";
  private transient int value3 = 3;
  BagOfPrimitives() { // no-args constructor }
}

BagOfPrimitives obj2 = gson.fromJson(json, BagOfPrimitives.class);
```

Moshi

- Modern **JSON library** for Android, Java and **Kotlin.**

```
BlackjackHand blackjackHand = new BlackjackHand(
    new Card('6', SPADES),
    Arrays.asList(new Card('4', CLUBS),
                  new Card('A', HEARTS)));

Moshi moshi = new Moshi.Builder().build();
JsonAdapter<BlackjackHand> jsonAdapter =
moshi.adapter(BlackjackHand.class);

String json = jsonAdapter.toJson(blackjackHand);
System.out.println(json);
```

```
val blackjackHand = BlackjackHand(
    Card('6', SPADES),
    listOf(Card('4', CLUBS), Card('A', HEARTS))
  )

val moshi: Moshi = Moshi.Builder().build()
val jsonAdapter: JsonAdapter<BlackjackHand> =
moshi.adapter<BlackjackHand>()

val json: String = jsonAdapter.toJson(blackjackHand)
println(json)
```

```
// https://github.com/square/moshi
implementation 'com.squareup.moshi:moshi:1.15.1'
```

Thank you for your attention

Mgr. Ing. Michal Krumnikl, Ph.D.

+420 597 325 867

michal.krumnikl@vsb.cz

www.vsb.cz