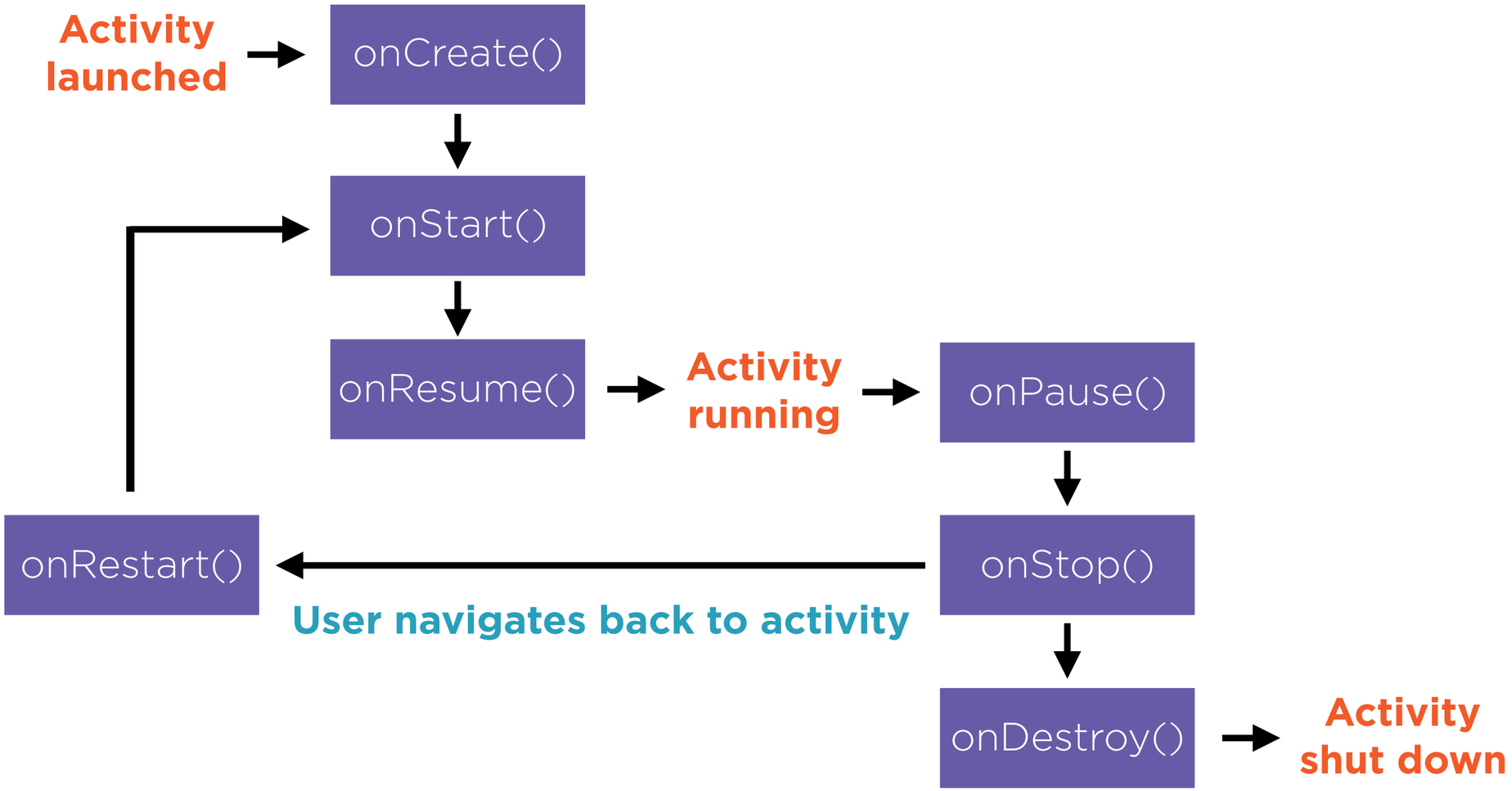
Implementing Life Cycle Aware Components

Omri Erez

SOFTWARE ENGINEER

@innovationMaze | https://www.linkedin.com/in/omrierez/

Simplified Activity Lifecycle



Application Components

- Bound to life cycle events

- Override life cycle callbacks

- Activity and service life cycle callbacks

The Reality

- Custom components often have life cycle related code

- Can result in long life cycle related methods

Bad implementation of life cycle dependent events can result in:

- Our application crashes

- Losing the user’s state and progress

- Not handling configuration changes gracefully

Not releasing resources properly can result in:

- Memory leaks

- Runtime crashes

- Inconsistency in application state

How can we handle these problems?

Solution
android.arch.lifecycle

- Make components be aware of life cycle events

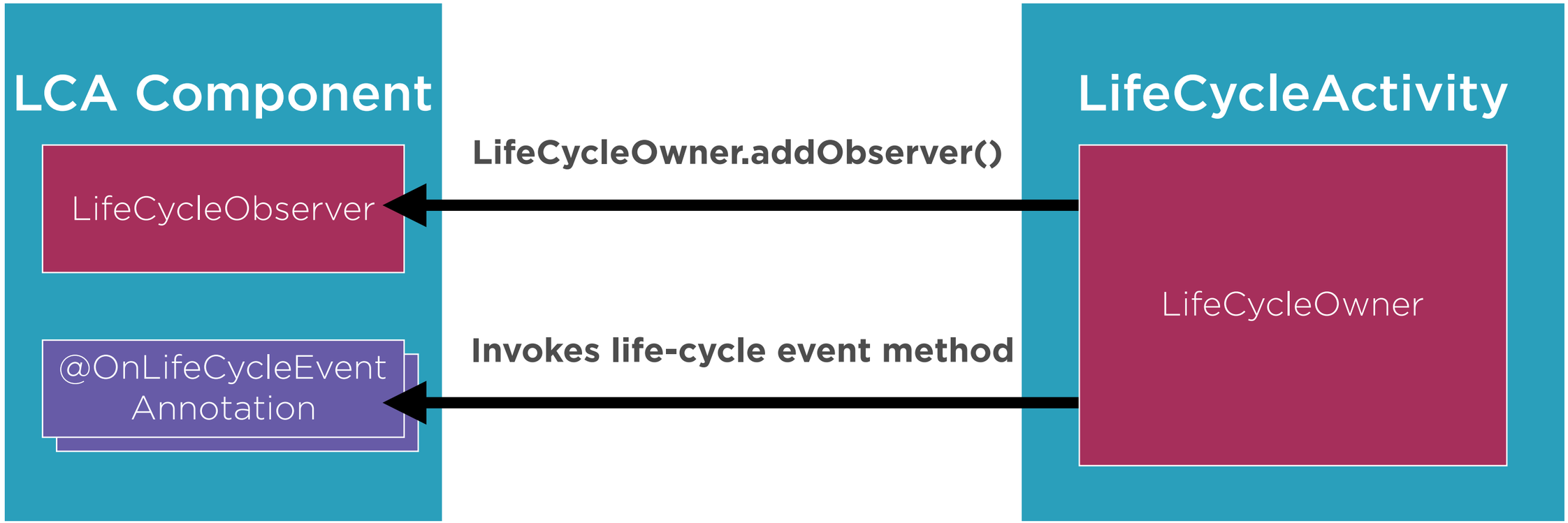
- Simple and powerful implementation

- Easy clearing of resources

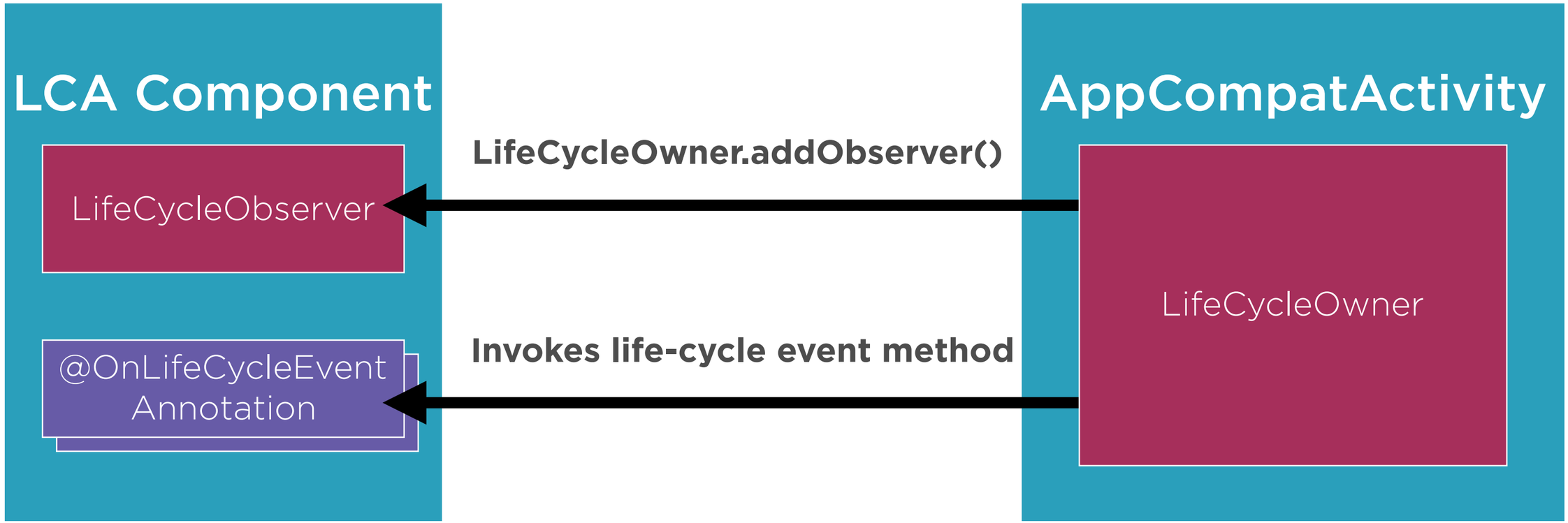
Using the life cycle aware apis will result in

- Decouple life cycle events' logic
- Shorten the activity life cycle methods
- Transfer the life cycle logic to be per component

LCA Prior Appcompat 26.1.0



LCA Appcompat 26.1.0 and Later



Demo

Crypto Boom App

- Identify components which have activity life cycle dependent logic
- Refactor them
- Implement LCA api’s
- Implement LCA logic

```
@Override
protected void onCreate(Bundle
savedInstanceState) {
    super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
    bindViews();
    fetchData();
    mTracker = new Tracker(this);
    mTracker.trackOnCreate();
    mFusedLocationClient =
LocationServices.getFusedLocationProvide
rClient(this);
    checkLocationPermission();
}
```

- Identify components:
  - Tracker
  - Location logic

DEMO REFACTOR TRACKERACTIVITY, LOCATIONACTIVITY & LCA COMPONENTS

```
@OnLifecycleEvent(Lifecycle.Event.ON_DESTROY)
public void trackOnDestroy() {
    Log.d(TAG, "trackOnDestroy() called");
    ((AppCompatActivity)mCon).getLifecycle().removeObserver(this);
    mQueue.add(generateTrackingStringRequest(“destroy"));

    Lifecycle.State currentState=((AppCompatActivity)mCon).getLifecycle().getCurrentState();

}
```

- Don’t forget to call removeObserver(this) on onDestroy()

- It’s possible to query the current life cycle event as well

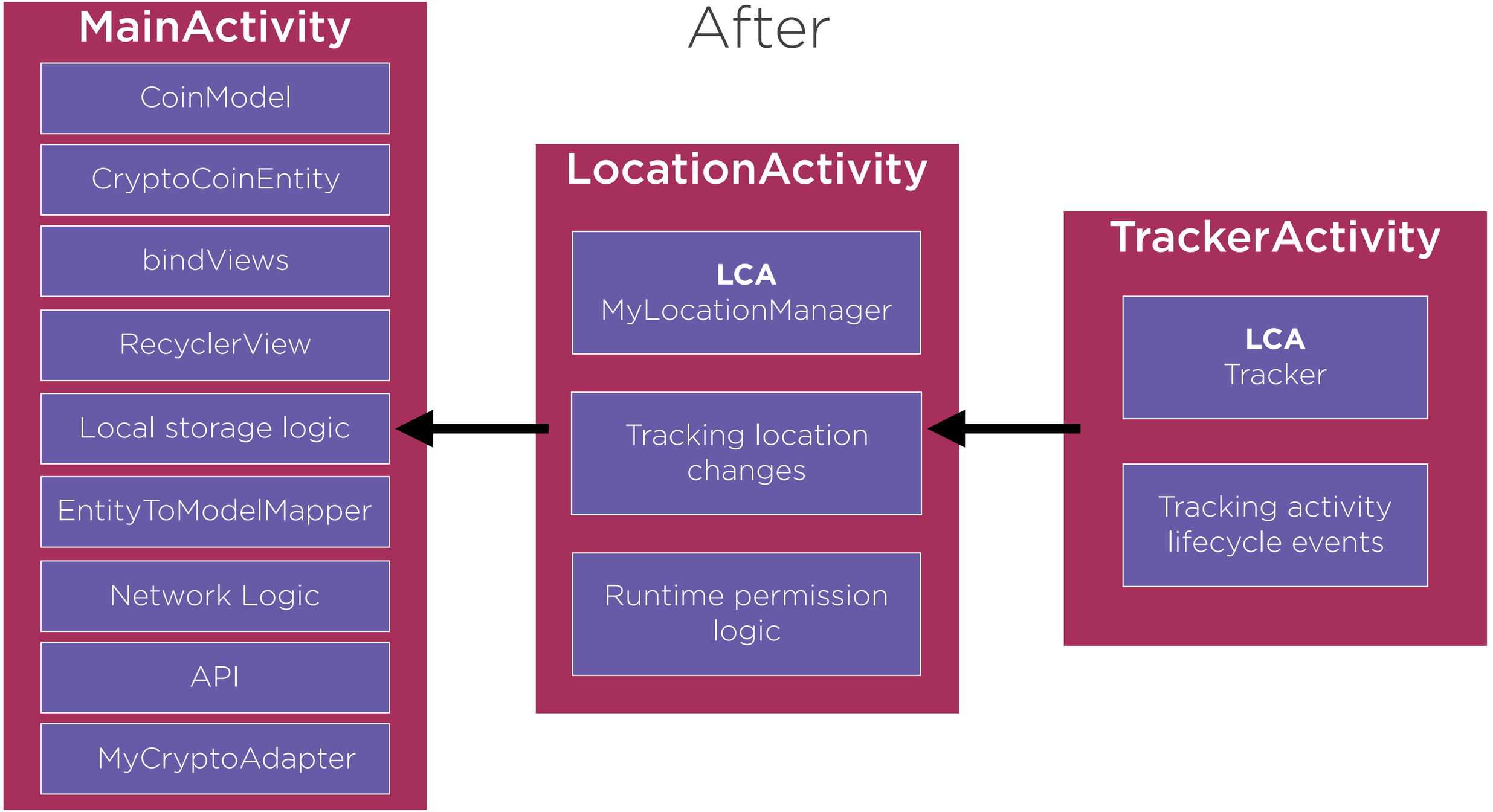
Overview of Changes

Before

Activity

- CoinModel
- CryptoCoinEntity
- Tracker:
  - Activity lifecycle
  - Location
- bindViews
- MyCryptoAdapter
- Network Logic for API request
- Persist data to local storage
- RecyclerView
- EntityToModelMapperTask
- Runtime permission logic
- Read data from local storage

After


MainActivity
CoinModel
CryptoCoinEntity
bindViews
RecyclerView
Local storage logic
EntityToModelMapper
Network Logic
API
MyCryptoAdapter
LocationActivity
LCA
MyLocationManager
Tracking location changes
Runtime permission logic
TrackerActivity
LCA
Tracker
Tracking activity lifecycle events

Summary

Life Cycle Aware Components

- Advantages:
  - Decouple LC logic per component
  - Makes activity LC methods to be shorter

```
@OnLifecycleEvent(Lifecycle.Event.ON_START)
public void trackOnStart() {
    Log.d(TAG, "trackOnStart() called");
    mQueue.add(generateTrackingStringRequest("start"));
}

@OnLifecycleEvent(Lifecycle.Event.ON_RESUME)
public void trackOnResume() {
    Log.d(TAG, "trackOnResume() called");
    mQueue.add(generateTrackingStringRequest("resume"));
}

@OnLifecycleEvent(Lifecycle.Event.ON_PAUSE)
public void trackOnPause() {
    Log.d(TAG, "trackOnPause() called");
    mQueue.add(generateTrackingStringRequest("pause"));
}
```

Summary

Additional Refactoring

- Broke down our God activity

- Added abstraction layers:
  - Tracking activity
  - Location activity