Android App Anatomy

Eric Burke
Square

@burke_eric

Topics

- Android lifecycle
- Fragments
- Open source
  - Tape
  - Otto
  - Dagger

1:28

Eric Burke

$12.00

One dozen cookies

Clear

1 2 3

4 5 6

7 8 9

00 0

1:32

Tipping

Collect Tips ON

Smart Tip Amounts OFF

CUSTOM TIP AMOUNTS

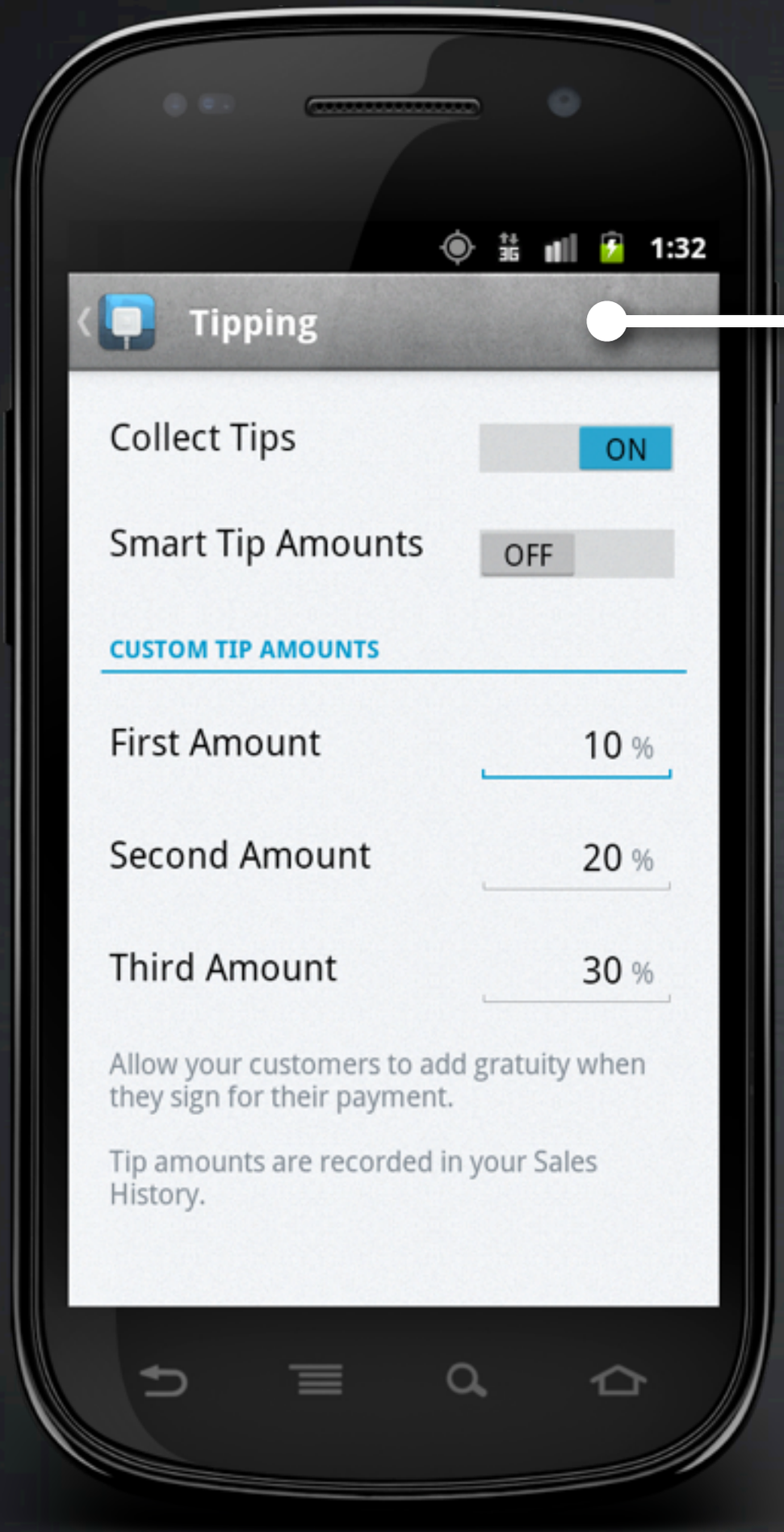
First Amount 10 %

Second Amount 20 %

Third Amount 30 %

Allow your customers to add gratuity when they sign for their payment.

Tip amounts are recorded in your Sales History.



ActionBarSherlock

Android Design on every device.

Lifecycle

Install

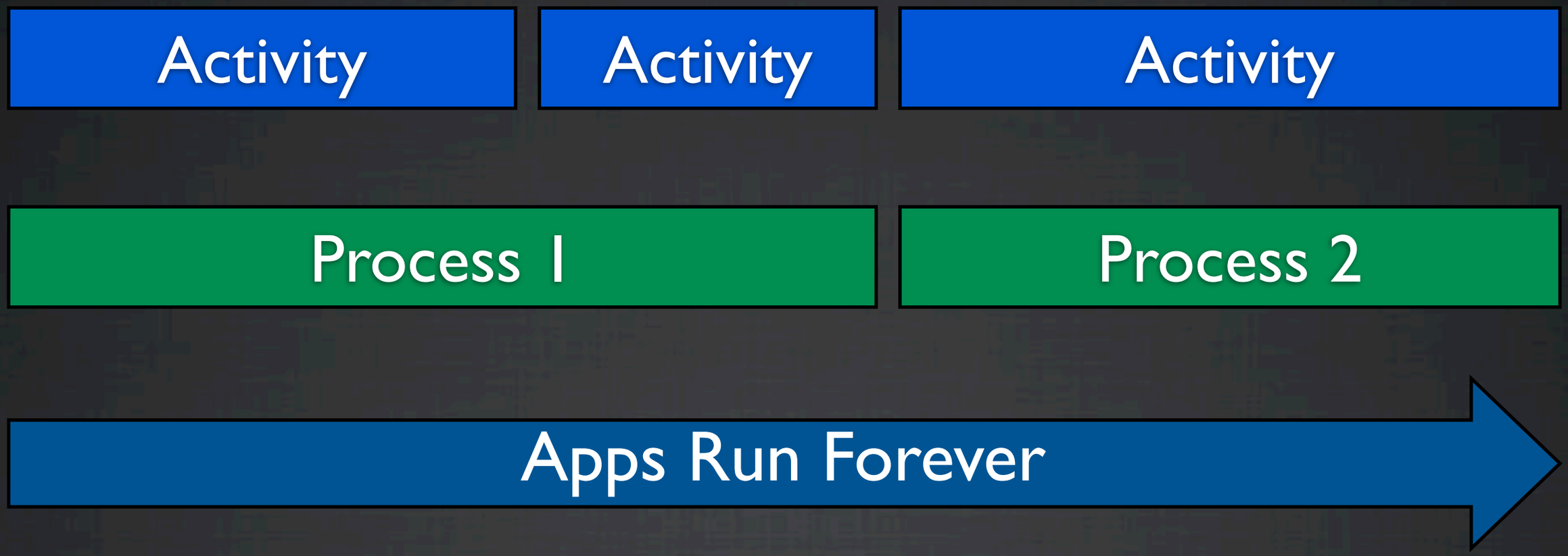
Apps Run Forever

Uninstall

Process 1

Process 2

Apps Run Forever


Activity
Activity
Activity
Process 1
Process 2
Apps Run Forever

R.I.P.

Static Variables

Activity

Activity

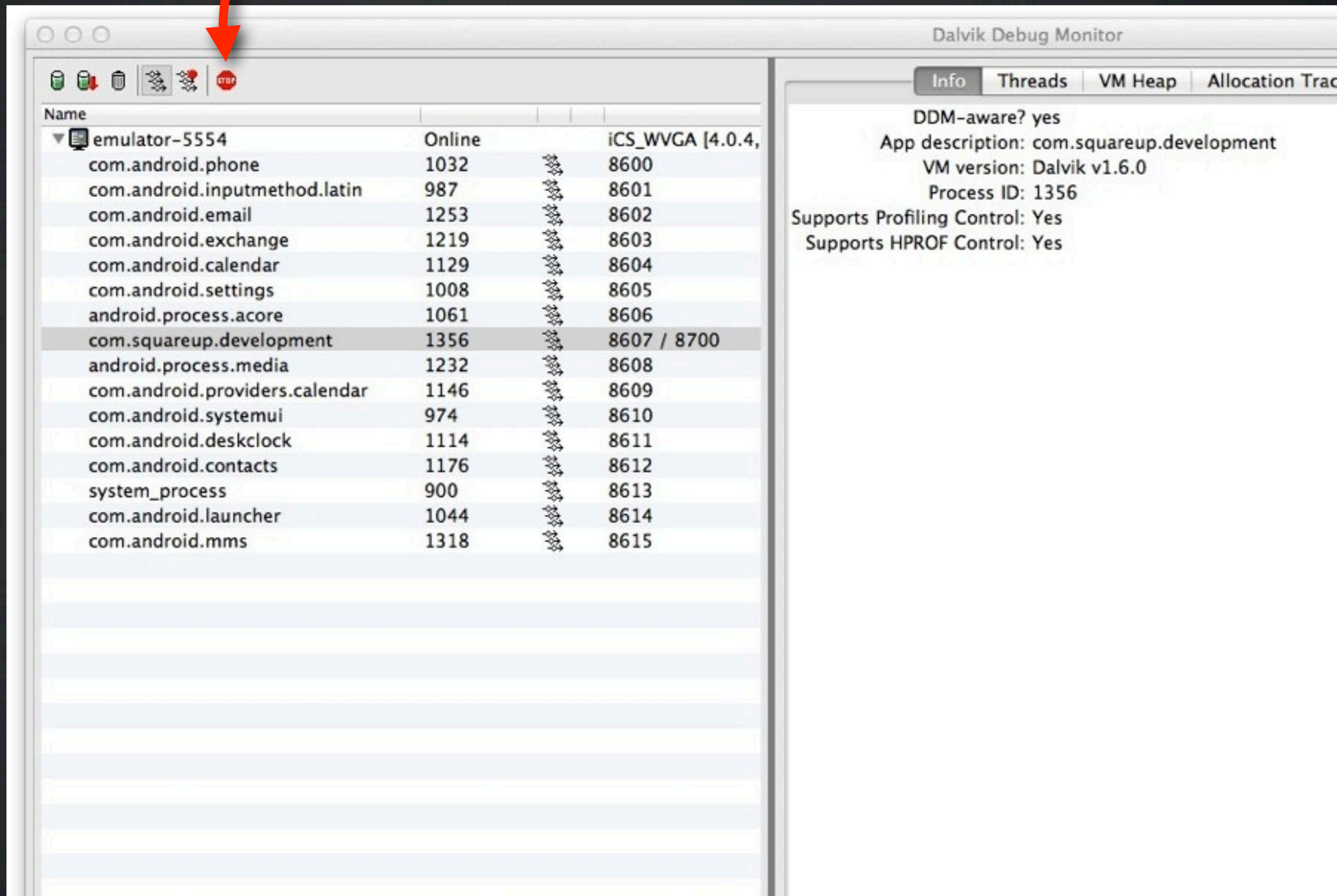
Activity

Process 1

Process 2

Apps Run Forever

Kill your process



Flash screen when apps do long operations on main thread

Show CPU usage

Screen overlay showing current CPU usage

Profile GPU rendering

Measure rendering time in adb shell dumpsys gfxinfo

Enable traces

No traces currently enabled

APPS

Don't keep activities

Destroy every activity as soon as the user leaves it

Background process limit

Standard limit

Background process limit

Standard limit

No background processes

At most 1 process

At most 2 processes

At most 3 processes

At most 4 processes

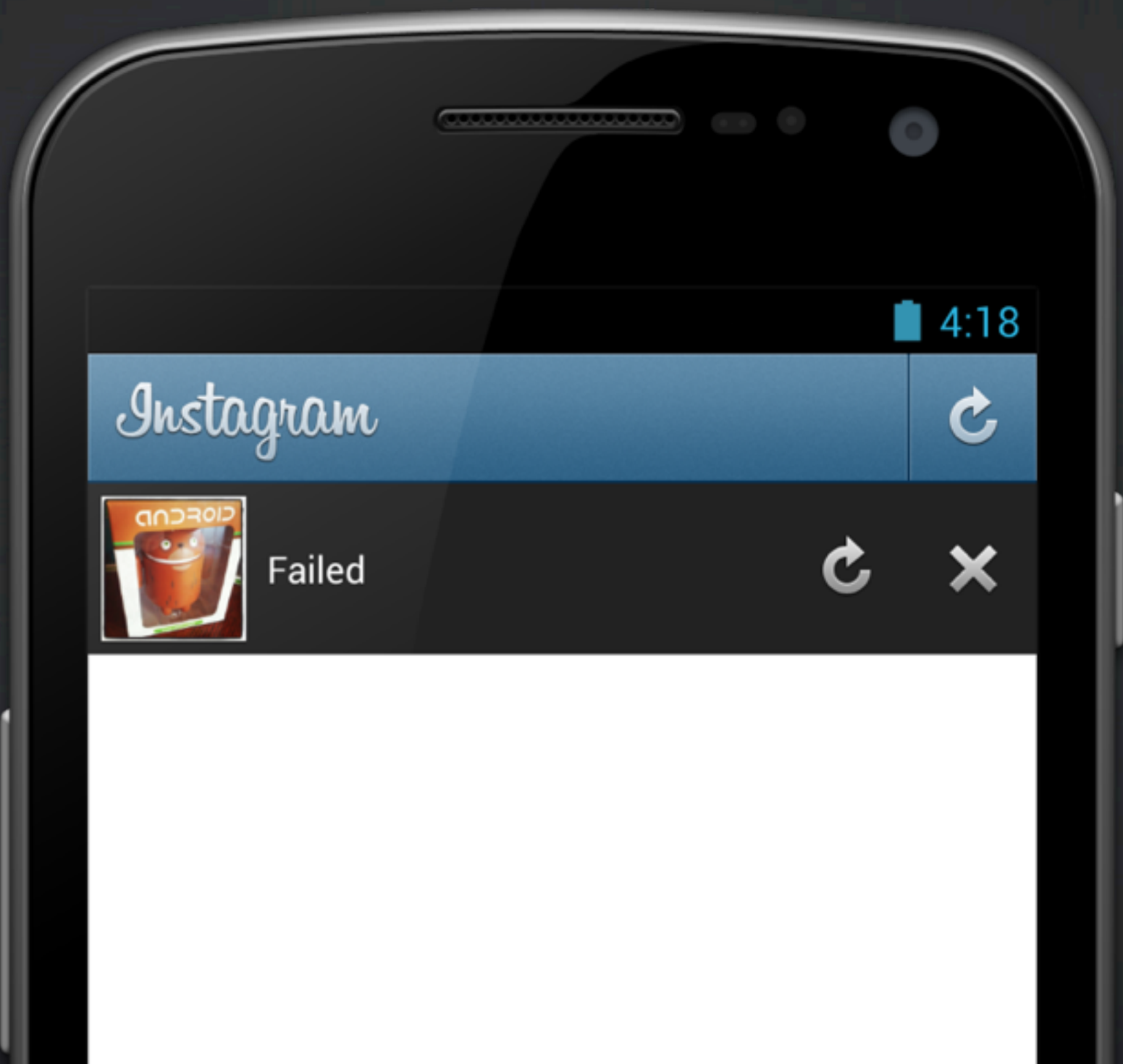
Cancel

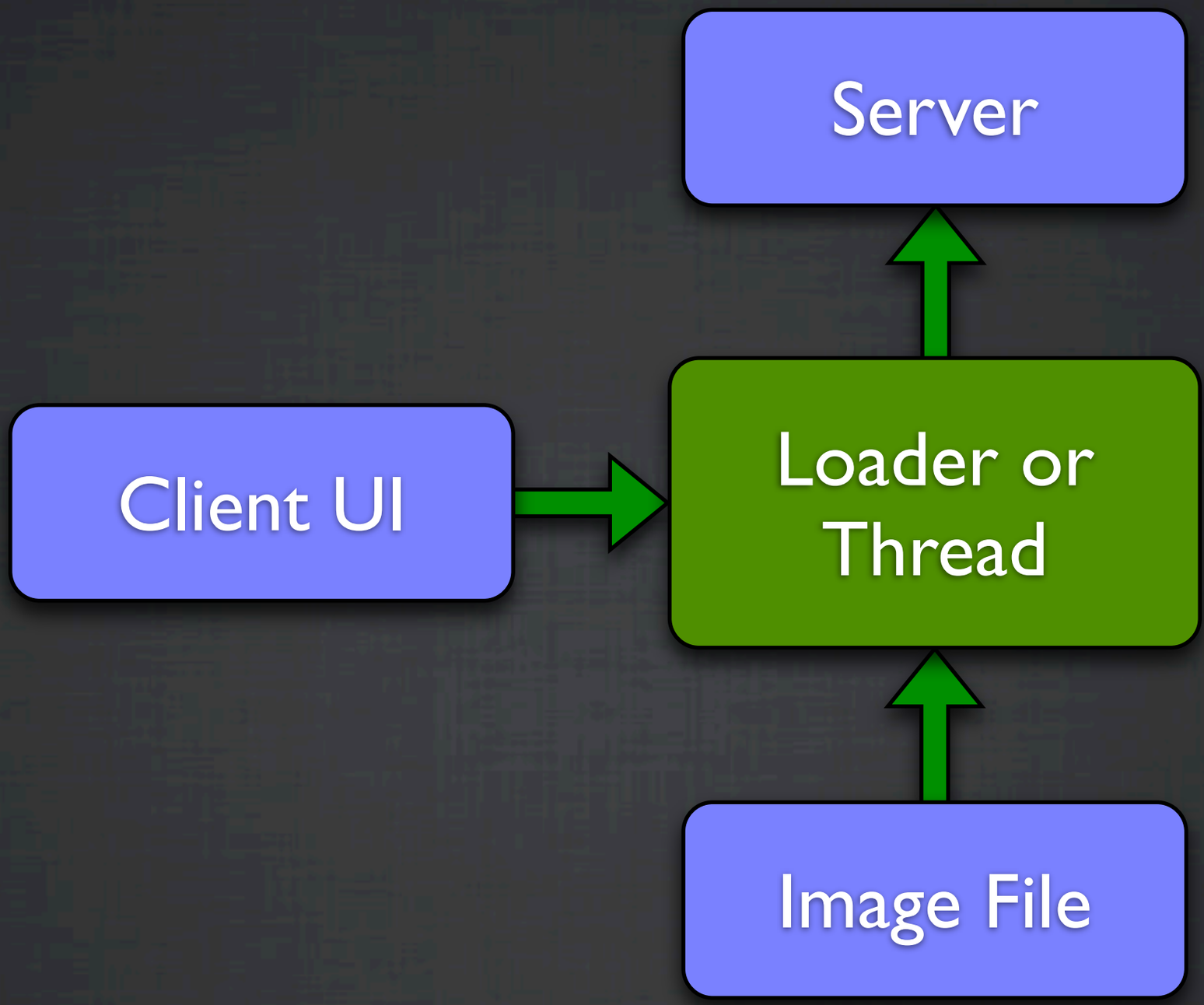
Tape

"If you respect users, persist tasks to disk."

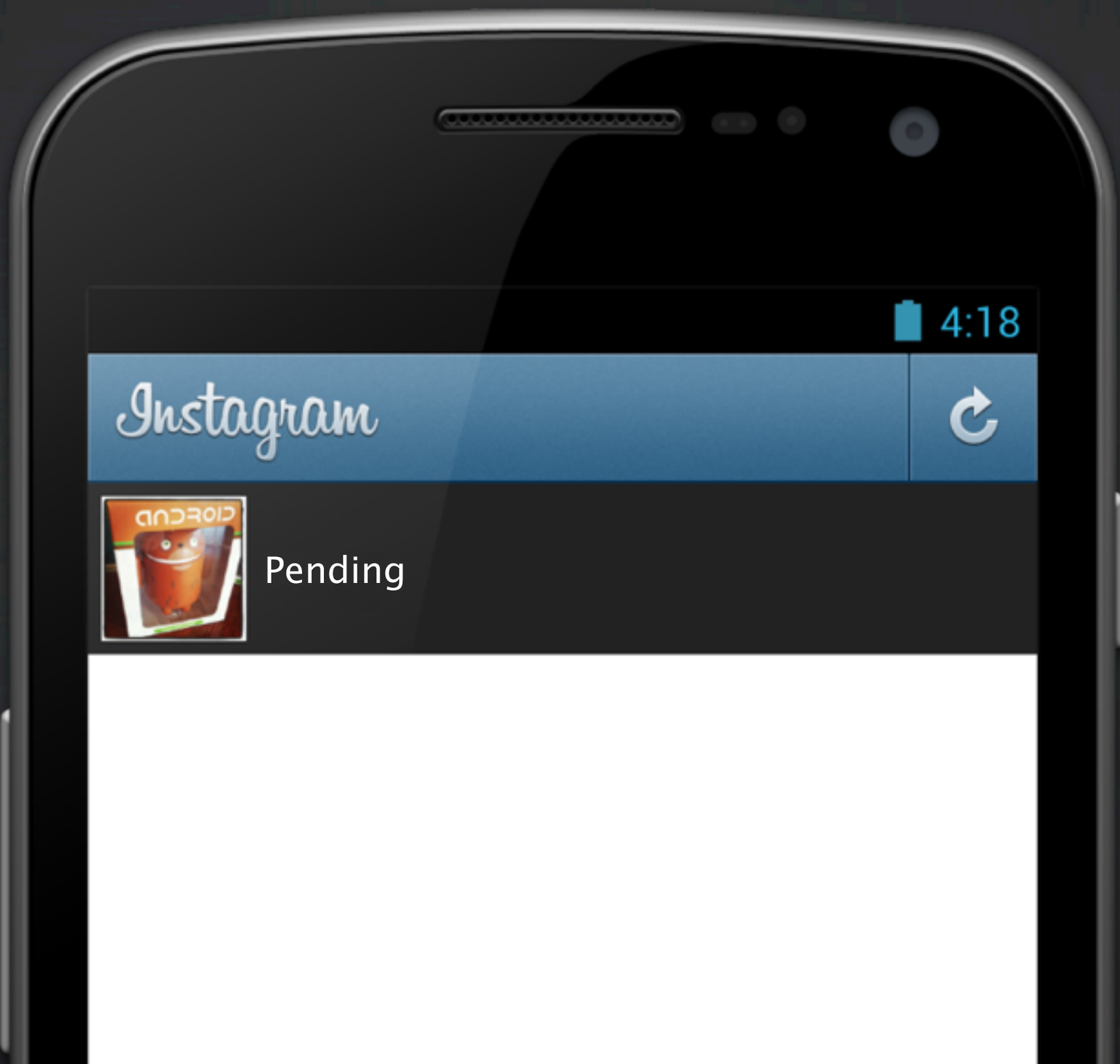
- Jesse Wilson

Don’t Do This




Server
Client UI
Loader or Thread
Image File

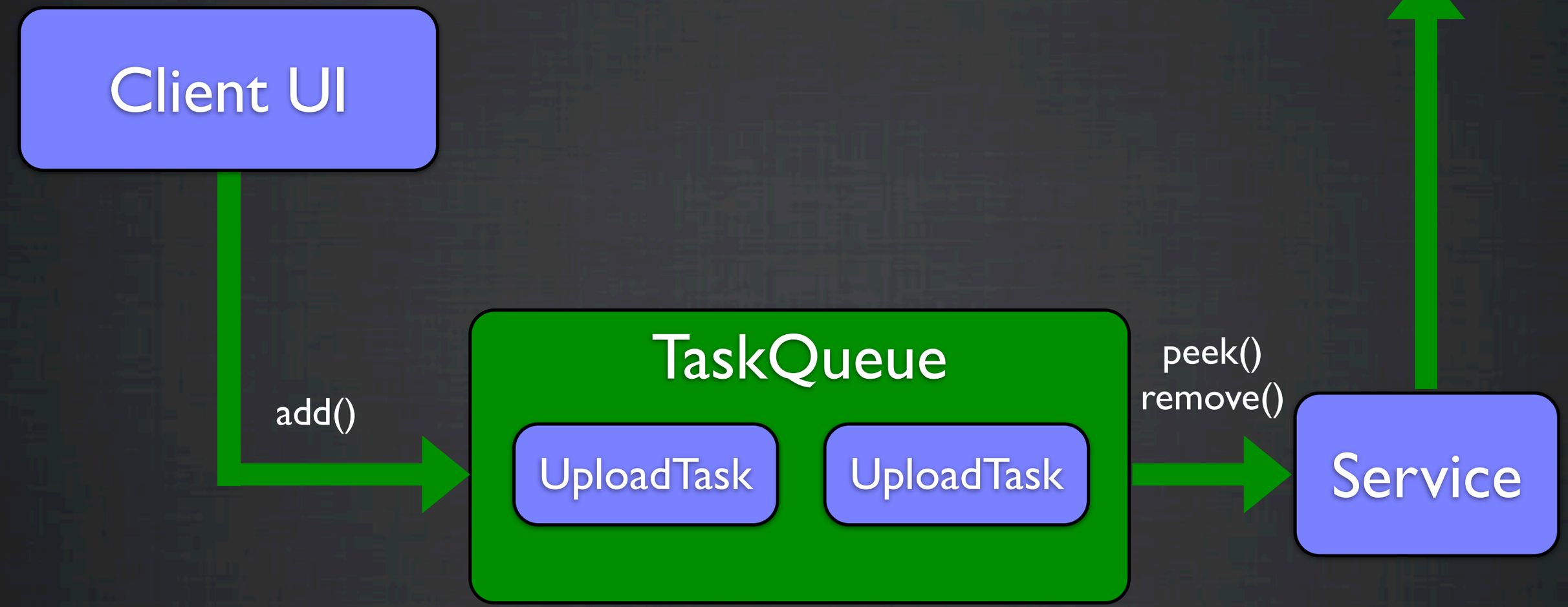
Do This



Tape API

- QueueFile
  - O(1) FIFO queue of `byte[]`
- ObjectQueue
  - A queue of <T>
- TaskQueue
  - Injects and starts tasks

Tape


Server
Client UI
add()
TaskQueue
UploadTask
UploadTask
peek()
remove()
Service

Usage Pattern

1. UI adds to a TaskQueue:

```
queue.add(new ImageUploadTask(image));
```

2. Start the Android Service:

```
context.startService(new Intent(context,
ImageUploadTaskService.class));
```

3. Service executes the next task:

```
Task t = queue.peek();
t.execute(this); // Listener
```

4. If successful:

```
queue.remove();
executeNext();
```

5. If failure, schedule a retry.

Retrying Failed Tasks

- Always process tasks in order: FIFO
- Distinguish between client bugs vs. network or server exceptions
  - Client errors – do not retry
  - Server & network errors – retry

Fragment Lifecycle

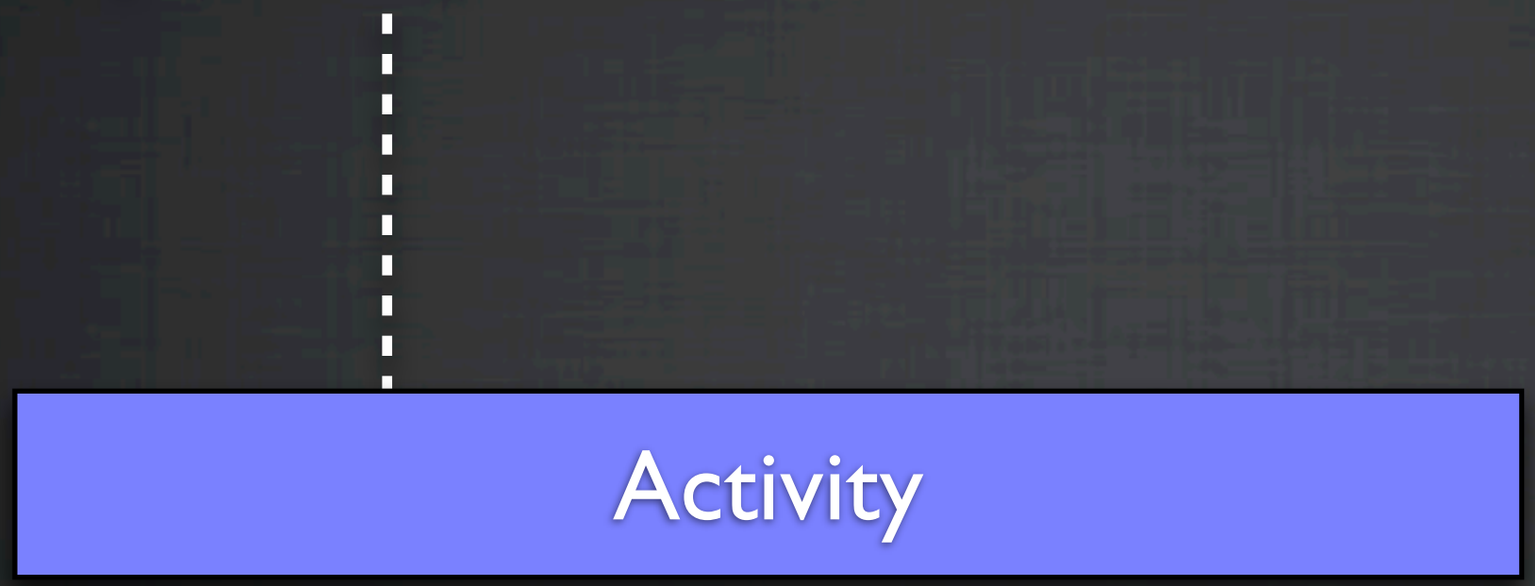
onCreate()

onSaveInstanceState()

Activity

Time →

Tap the Upload Button

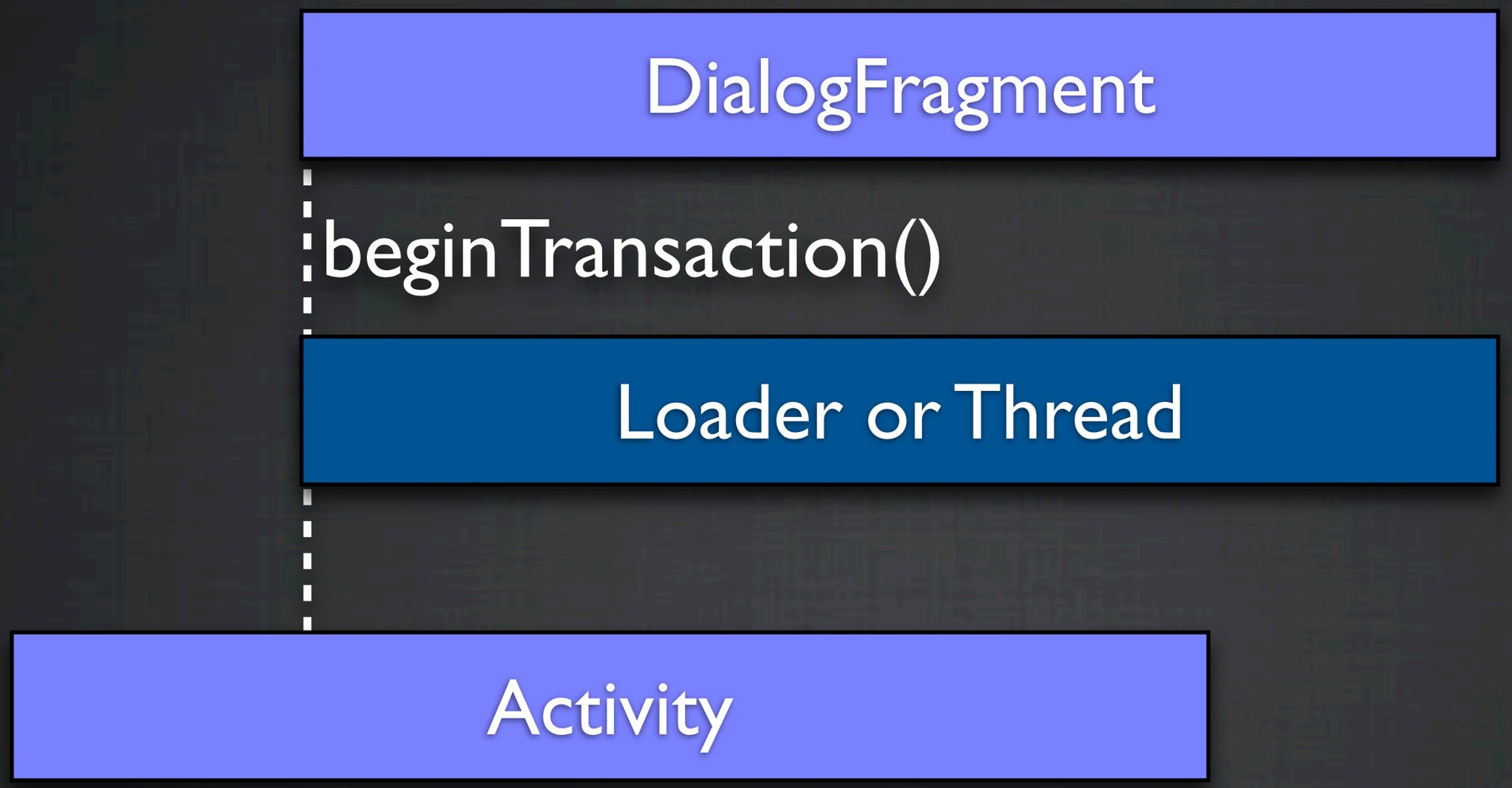


Time →

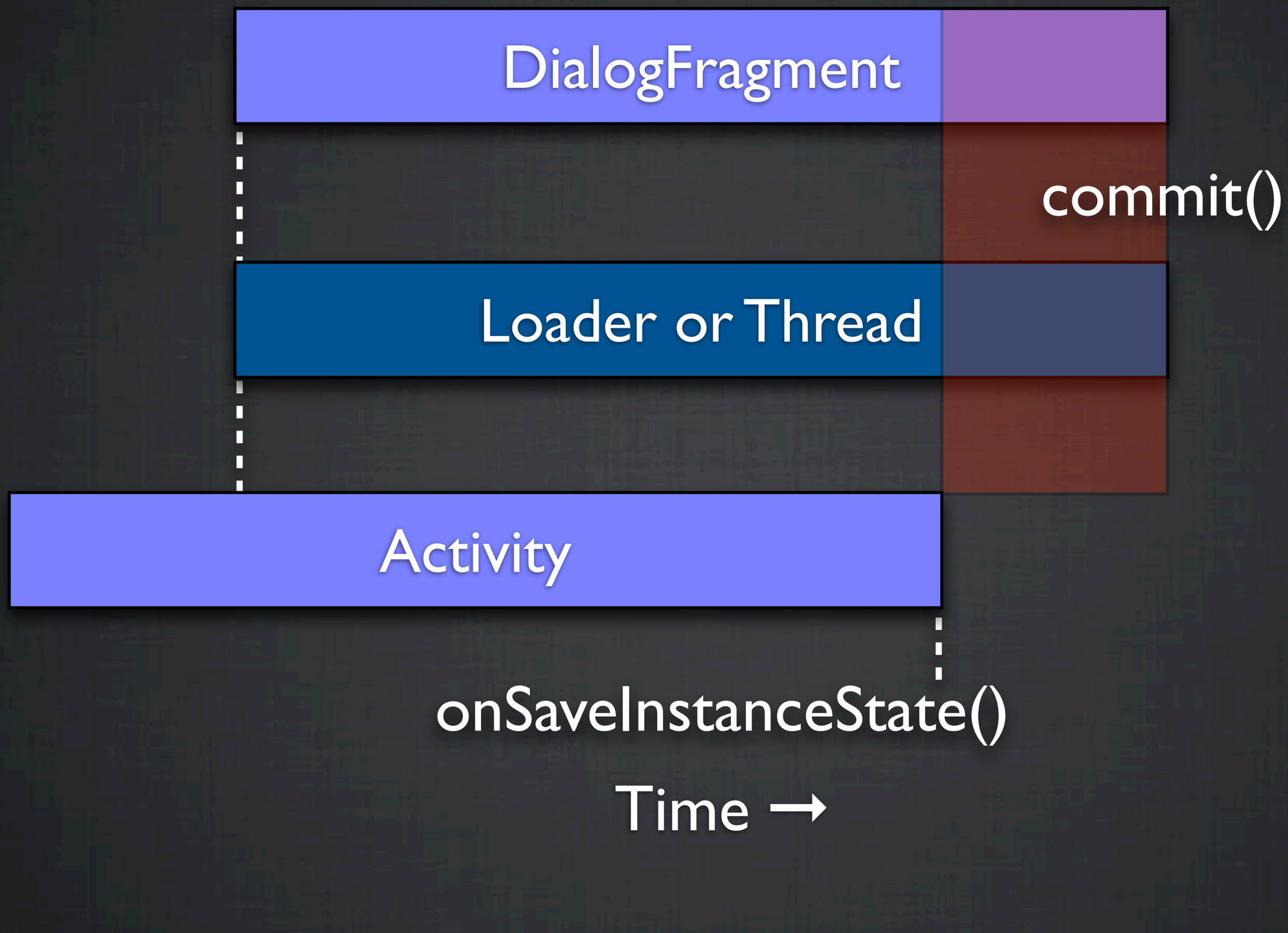
Loader or Thread

Activity

Time →


DialogFragment
beginTransaction()
Loader or Thread
Activity

Time →


DialogFragment
commit()
Loader or Thread
Activity
onSaveInstanceState()
Time →

Square does not* use Loaders.

Fragment Layout

12:44

Personal Info

Your personal information

We keep this information secure and use it to verify your identity.

YOUR LEGAL NAME

Eric Burke

YOUR HOME ADDRESS

Street

Apt / Suite

City

State

ZIP

DATE OF BIRTH

mm / dd / yyyy

LAST FOUR DIGITS OF SOCIAL SECURITY NUMBER

12:44

Bank Account

Where should we deposit your money?

NAME ON BANK ACCOUNT

Eric Burke

BANK ACCOUNT INFORMATION

Routing Number

Account Number

Confirm Account Number

Advantages

- Smooth animations between steps
- ActionBar does not move
- Code organization

Onboarding

Fragment

Fragment

Fragment

Settings

Fragment

Fragment

Fragment

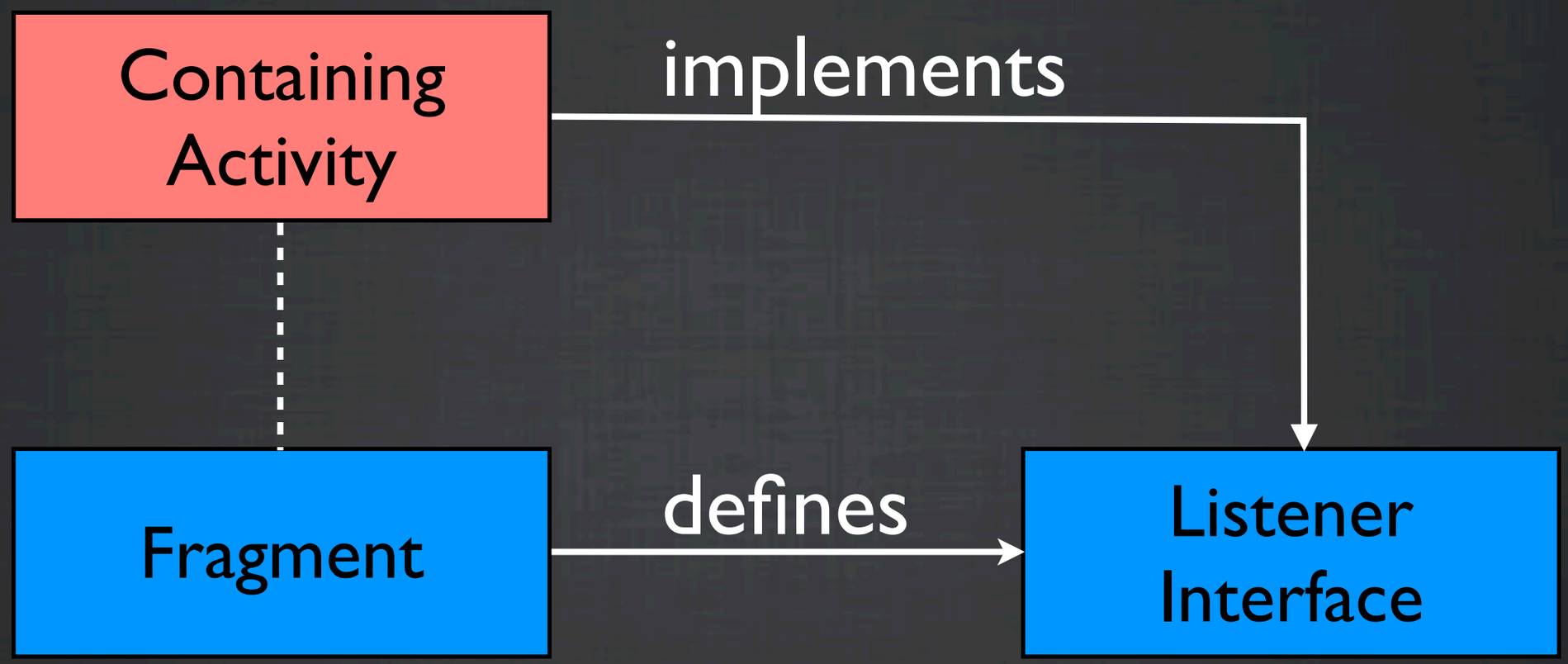
Payment Flow

Fragment

Fragment

Fragment

Fragment → Activity Communication



http://developer.android.com/training/basics/fragments/communicating.html

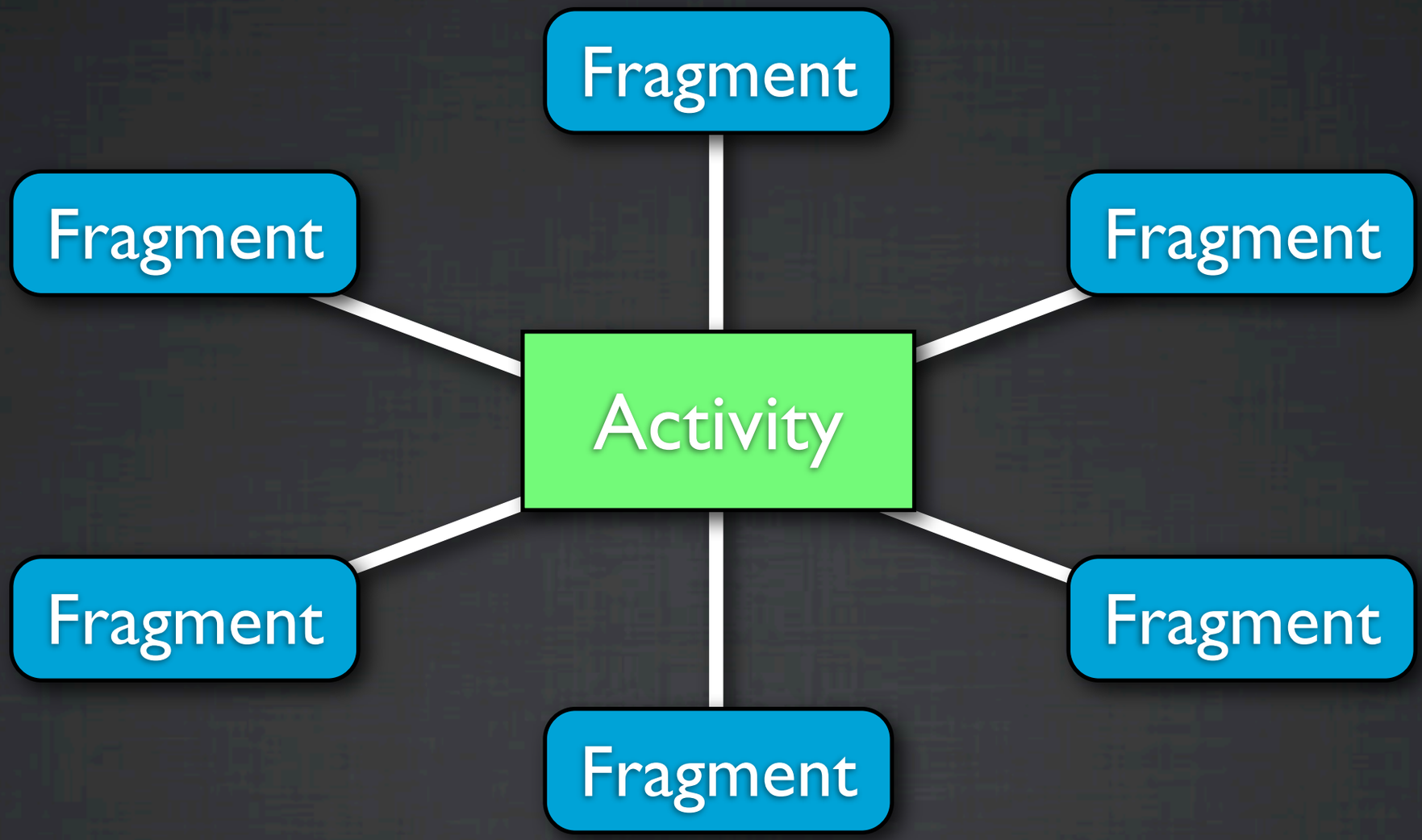
Listener Interface

```
public class NewsFragment extends Fragment {
  private Listener listener;

  public interface Listener {
    void onItemSelected(int position);
  }

  @Override public void onAttach(Activity a) {
    super.onAttach(a);

    listener = (Listener) a;
  }
}
```

Activity

```
public class HomeActivity extends Activity
        implements NewsFragment.Listener {

    @Override
    public void onItemSelected(int position) {
        …
    }
}
```

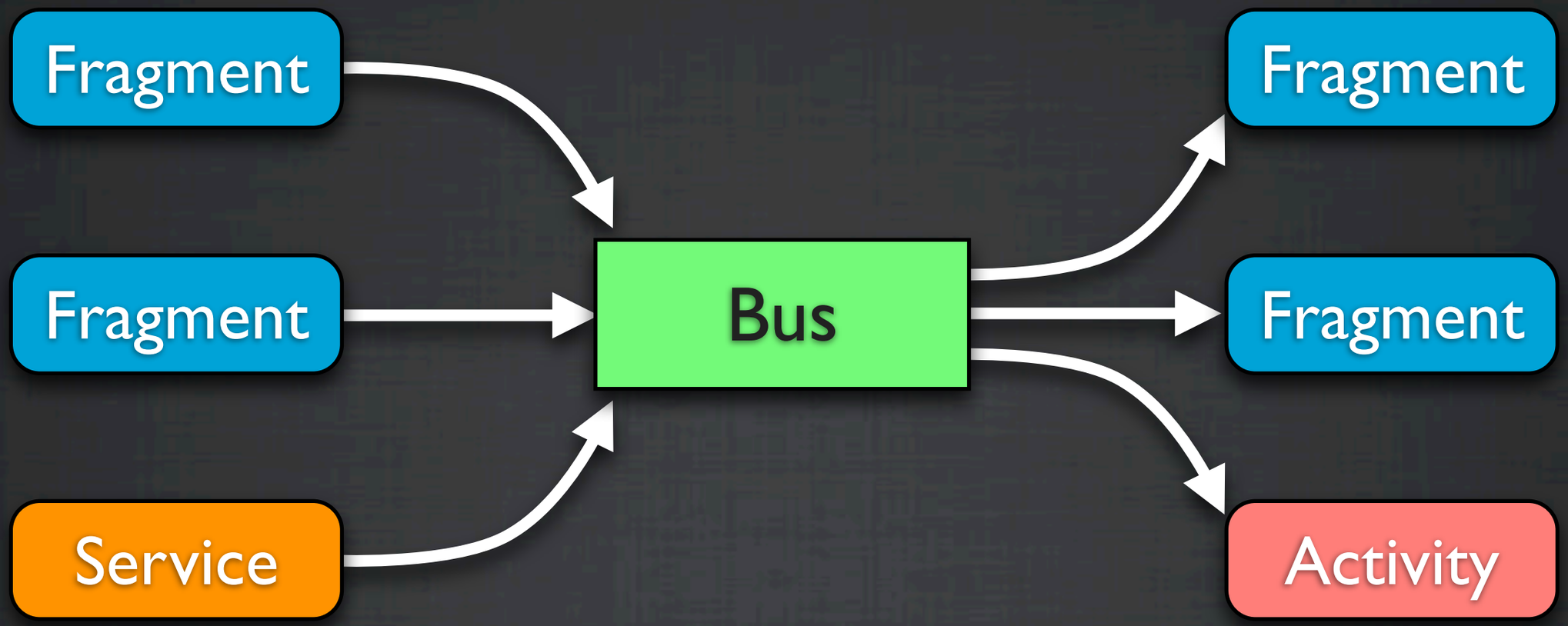
```
public class Onboarding extends Activity
   implements AccountFragment.Listener,
              ActivateFragment.Listener,
              ShippingFragment.Listener,
              BankFragment.Listener,
                 etc...
{
   // The god object...
}
```



How can Fragments communicate without tight Activity coupling?

Publish

Subscribe



LocalBroadcastManager

- Included in Android Support library
- Publishes Intents within your process

Publishing

```
Intent intent = new Intent(
    BroadcastIds.LOCATION_UPDATED_ACTION);

intent.putExtra(
    BroadcastIds.CURRENT_LOCATION_KEY,
    getCurrentLocation());

LocalBroadcastManager.getInstance(
    getActivity()).sendBroadcast(intent);
```

Publishing

```
Intent intent = new Intent(
    BroadcastIds.LOCATION_UPDATED_ACTION);

intent.putExtra(
    BroadcastIds.CURRENT_LOCATION_KEY,
    getCurrentLocation());

LocalBroadcastManager.getInstance(
    getActivity()).sendBroadcast(intent);
```

Publishing

```
Intent intent = new Intent(
    BroadcastIds.LOCATION_UPDATED_ACTION);

intent.putExtra(
    BroadcastIds.CURRENT_LOCATION_KEY,
    getCurrentLocation());

LocalBroadcastManager.getInstance(
    getActivity()).sendBroadcast(intent);
```

Subscribing

```
BroadcastReceiver receiver = new BroadcastReceiver() {
   @Override public void onReceive(
         Context context, Intent intent) {

      Location location =
            (Location) intent.getParcelableExtra(
                  BroadcastIds.CURRENT_LOCATION_KEY);

      showLocation(location);
   }
};
```

Subscribing

```
IntentFilter locationFilter = new IntentFilter(
    BroadcastIds.LOCATION_UPDATED_ACTION);


@Override public void onResume() {
  super.onResume();
  LocalBroadcastManager.getInstance(getActivity())
      .registerReceiver(receiver, locationFilter);
}

@Override public void onPause() {
  super.onPause();
  LocalBroadcastManager.getInstance(getActivity())
      .unregisterReceiver(receiver);
}
```

Boilerplate.
No type safety.
Hard to test.

Otto

Registration

```
public class BaseFragment extends Fragment {
  @Inject Bus bus;

  @Override public void onResume() {
    super.onResume();
    bus.register(this);
  }

  @Override public void onPause() {
    super.onPause();
    bus.unregister(this);
  }
}
```

Registration

```
public class BaseFragment extends Fragment {
  @Inject Bus bus;

  @Override public void onResume() {
    super.onResume();
    bus.register(this);
  }

  @Override public void onPause() {
    super.onPause();
    bus.unregister(this);
  }
}
```

Fails fast at runtime.

Subscribing

```
@Subscribe
public void onLocationUpdated(Location l) {
   showLocation(l);
}
```

Receives any type
extending Location.

Publishing

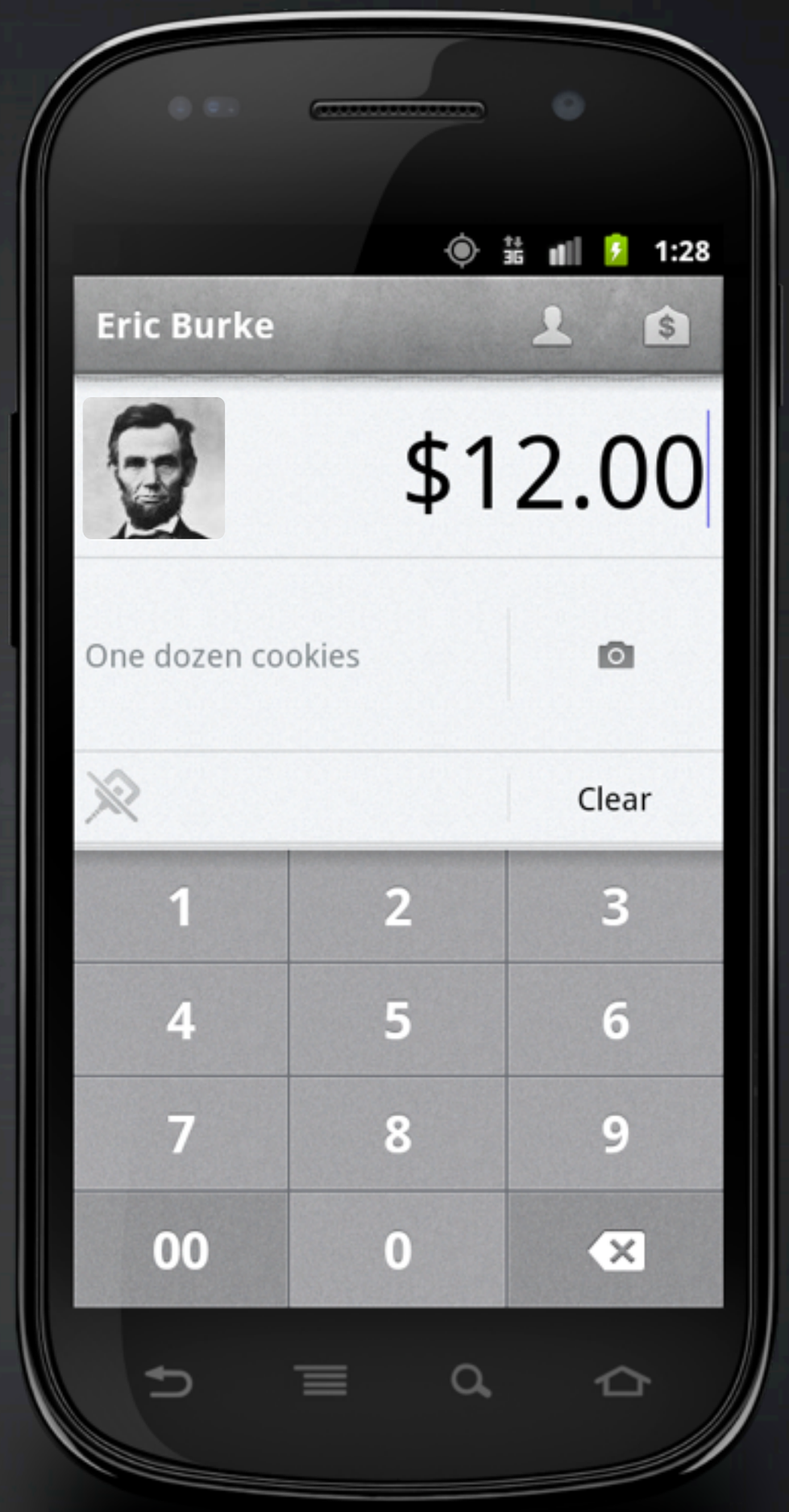
```
bus.post(getCurrentLocation());
```

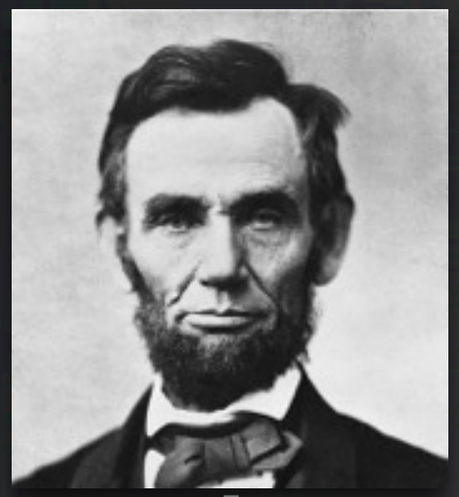
Synchronous delivery.

@Produce

(getting data the first time)

How to get this image?





Downloader

post(UserImage)

Bus

UserImageCache

Subscribing

```
public class AccountActivity
    extends BaseActivity {

  @Subscribe public void onUserImageUpdated(
      UserImage image) {
    ((ImageView) findViewById(R.id.image))
        .setImageBitmap(image.getBitmap());
  }
```

get()

done()

onResume()

onPause()

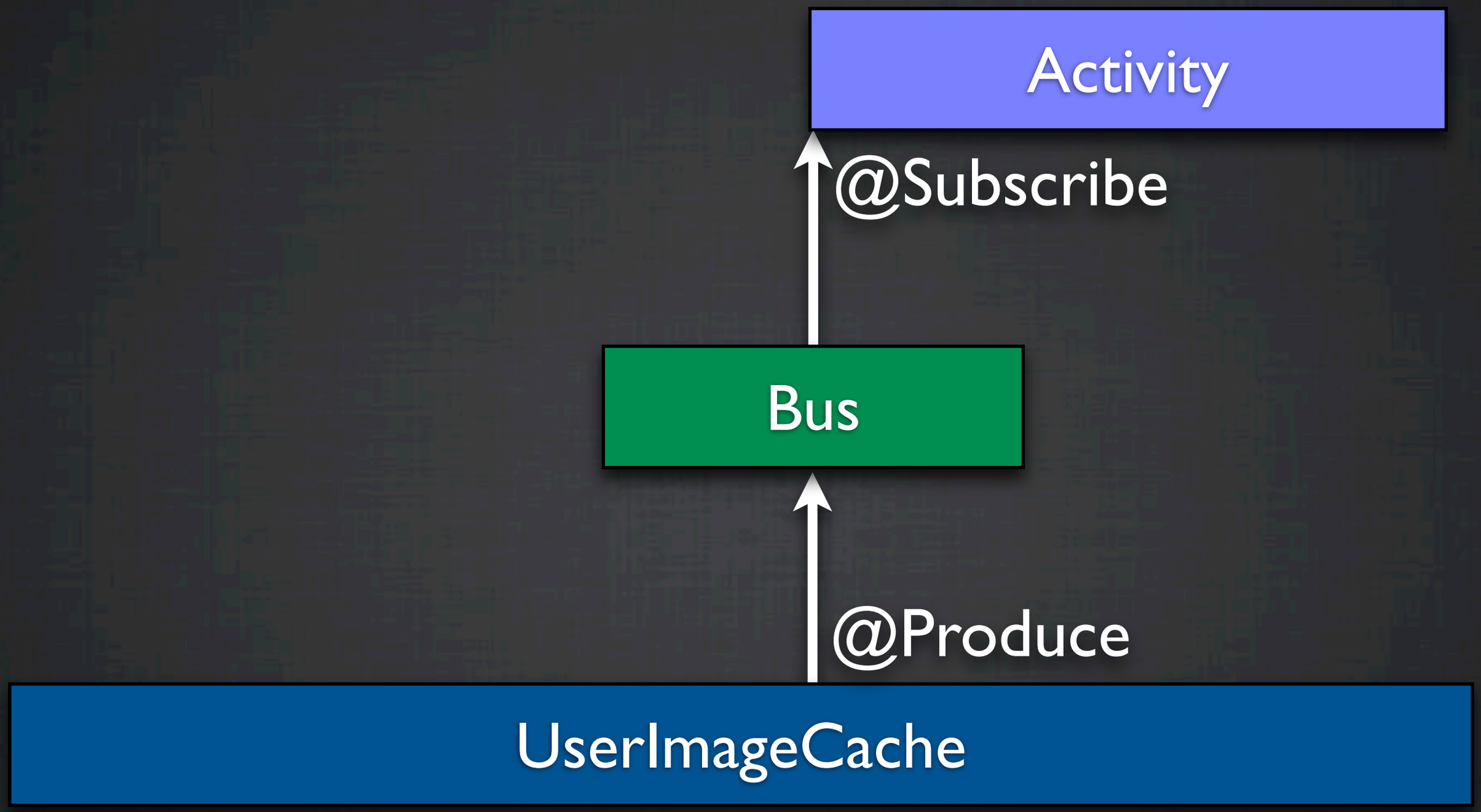
Downloader

Activity

UserImageCache

Producers

```
@Singleton public class UserImageCache {

   @Produce
   public UserImage produceUserImage() {
      return cachedUserImage;
   }

   …
}
```


Activity
@Subscribe
Bus
@Produce
UserImageCache

`@Produce` decouples threads from the Activity and Fragment lifecycle.

Otto API Summary

- register(), unregister(), post()
- @Subscribe, @Produce
- Thread confinement
- Easy to test!

Origins of Otto

- Forked from Guava's EventBus
- Optimized for Android – 16k!
- Less reflection; more caching

Dependency Injection

Guice on Android?

- We've used it for 2+ years
- Startup performance is a challenge
- Runtime error checking
- See also: RoboGuice

Can we do better?

† Dagger

What is Dagger?

Compile-time
dependency injection.

†

@Inject

```
import javax.inject.Inject; // JSR-330

public class PublishFragment
    extends BaseFragment {

  @Inject Bus bus;
  @Inject LocationManager locationManager;

  …
}
```

† Constructor Injection

```
public class AccountUpdater {

  private final Bus bus;
  private final AccountService accounts;

  @Inject public AccountUpdater(Bus bus,
      AccountService accounts) {
    this.bus = bus;
    this.accounts = accounts;
  }
}
```

Module

```
@Module(
    entryPoints = {
        HomeActivity.class,
        PublishFragment.class,
        SubscribeFragment.class
    }
)
public class ExampleModule {

  @Provides @Singleton Bus provideBus() {
    return new Bus();
  }
}
```

Module

```
@Module(
    entryPoints = {
        HomeActivity.class,
        PublishFragment.class,
        SubscribeFragment.class
    }
)
public class ExampleModule {

  @Provides @Singleton Bus provideBus() {
    return new Bus();
  }
}
```

Missing Provider Method?

No injectable members on com.squareup.otto.Bus.
Do you want to add an injectable constructor?
required by com.squareup.anatomy.PublishFragment
for com.squareup.anatomy.ExampleModule

Bootstrapping

```
public class ExampleApp extends Application {
  private ObjectGraph objectGraph;

  @Override public void onCreate() {
    super.onCreate();

    objectGraph = ObjectGraph.get(
        new ExampleModule(this));
  }

  public ObjectGraph objectGraph() {
    return objectGraph;
  }
}
```

Bootstrapping

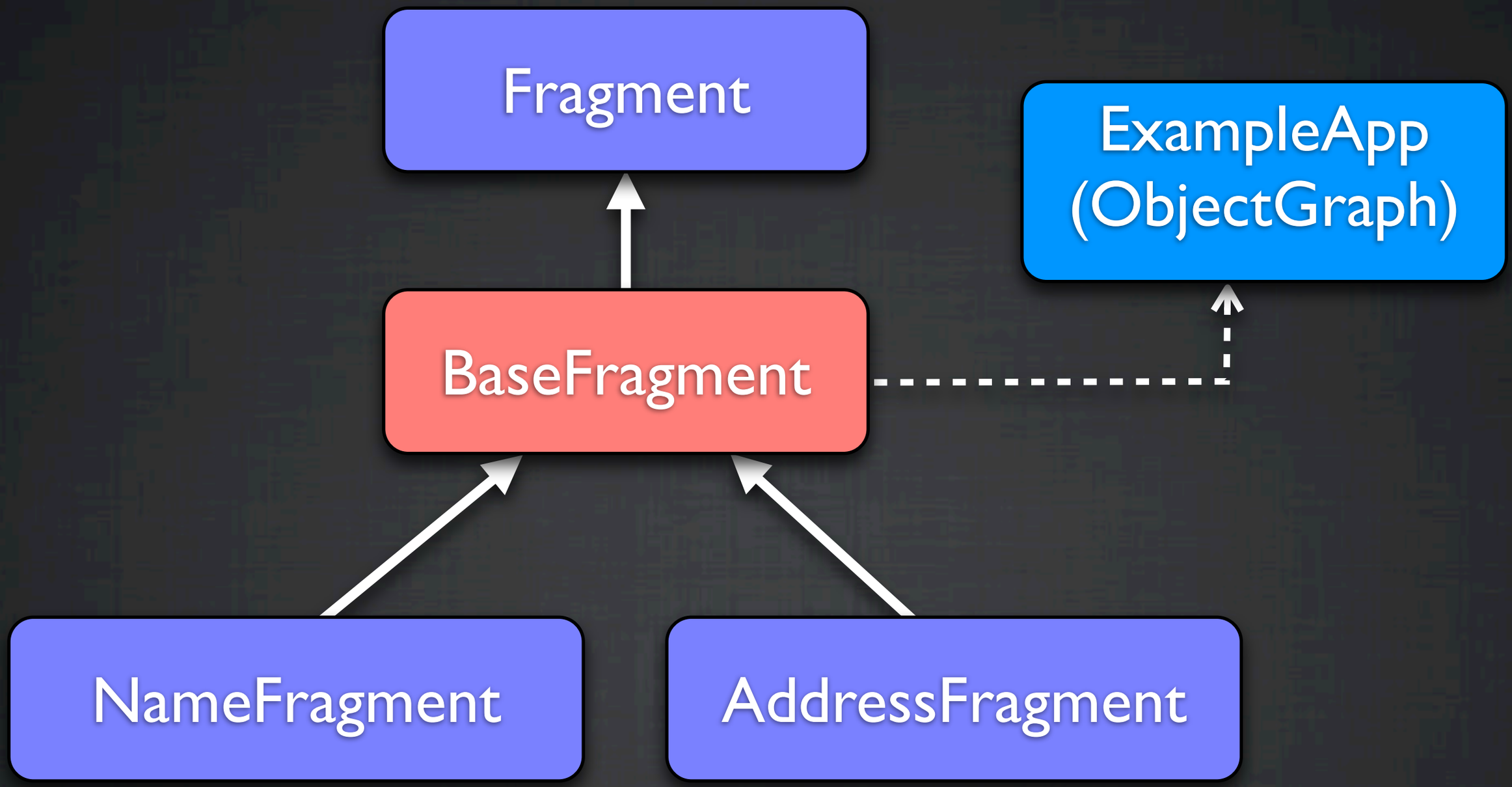
```
public class ExampleApp extends Application {
  private ObjectGraph objectGraph;

  @Override public void onCreate() {
    super.onCreate();

    objectGraph = ObjectGraph.get(
        new ExampleModule(this));
  }

  public ObjectGraph objectGraph() {
    return objectGraph;
  }
}
```

Bootstrapping

```
public class ExampleApp extends Application {
  private ObjectGraph objectGraph;

  @Override public void onCreate() {
    super.onCreate();

    objectGraph = ObjectGraph.get(
        new ExampleModule(this));
  }

  public ObjectGraph objectGraph() {
    return objectGraph;
  }
}
```

Bootstrapping

```
public class ExampleApp extends Application {
  private ObjectGraph objectGraph;

  @Override public void onCreate() {
    super.onCreate();

    objectGraph = ObjectGraph.get(
        new ExampleModule(this));
  }

  public ObjectGraph objectGraph() {
    return objectGraph;
  }
}
```


Fragment
ExampleApp
(ObjectGraph)
BaseFragment
NameFragment
AddressFragment

Fragment Injection

```
public class BaseFragment extends Fragment {

   @Override
   public void onCreate(Bundle state) {
      super.onCreate(state);

      ((ExampleApp) getActivity()
            .getApplication())
            .objectGraph().inject(this);
   }
}
```

Fragment Injection

```
public class BaseFragment extends Fragment {

   @Override
   public void onCreate(Bundle state) {
      super.onCreate(state);

      ((ExampleApp) getActivity()
          .getApplication())
          .objectGraph().inject(this);
   }
}
```

Fragment Injection

```
public class BaseFragment extends Fragment {

   @Override
   public void onCreate(Bundle state) {
      super.onCreate(state);

      ((ExampleApp) getActivity()
            .getApplication())
            .objectGraph().inject(this);
   }
}
```

Dagger Limitations

- No final and private field injection
- No method injection
- No scopes
- @Assisted

Dagger Features

- Compile time injection
- Very little magic
- über fast

https://github.com/eburke/presentations

Tape → square.github.com/tape/

Otto → square.github.com/otto/

ActionBarSherlock → actionbarsherlock.com

Dagger → square.github.com/dagger