# TOOTHPICK

## A FRESH APPROACH TO DI

https://github.com/stephanenicolas/toothpick/

# ABOUT US

## DANIEL MOLINERO REGUERA


Android Dev @ Groupon

Best level 3 ever

OSS: Dart,

## STEPHANE NICOLAS


Senior Android Dev @ Groupon

~20 years of Java coding

OSS: Dart, TP

GROUPON

# The Android Team is Hiring :

jobs.groupon.com/careers/

# THE TALK

## WHY DI ?

## WHY TOOTHPICK?

## TOOTHPICK FEATURES

- SCOPE
- MODULES & BINDINGS
- SCOPE TREE
- SCOPE SINGLETONS

## ADVANCED FEATURES

- MVP & STATE PRESERVATION
- MULTIPLE ACTIVITIES FLOWS

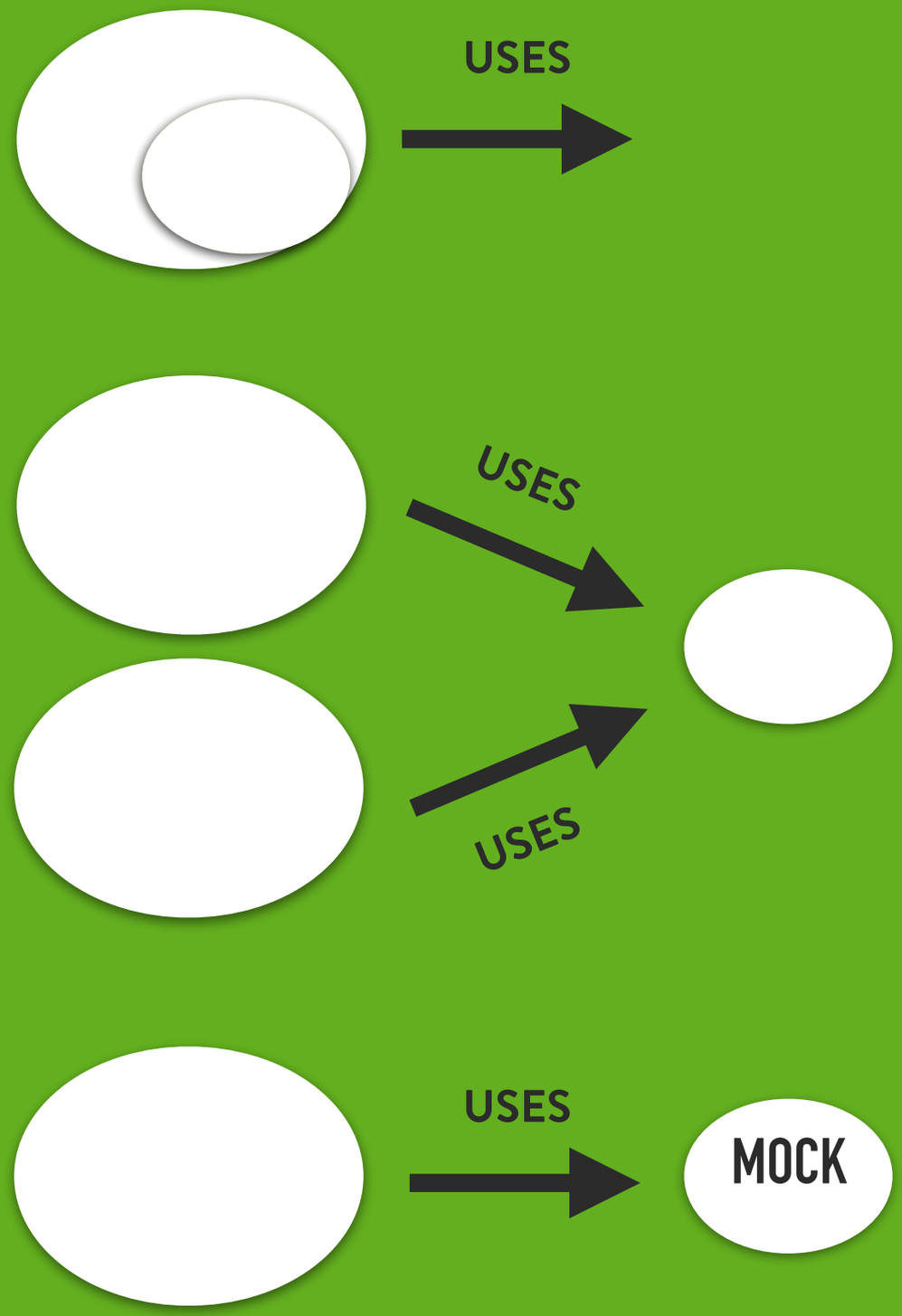
## CONCLUSION

# WHY DI ?

DECOUPLE

REUSE

TEST



# WHY TOOTHPICK?

SIMPLER

BETTER TEST SUPPORT

EVEN FASTER

# TOOTHPICK FEATURES

```
public class SmoothieMachine {

    @Inject IceMachine iceMachine;

    public void doSmoothie() {
        iceMachine.makeIce();
    }
}
```

```
@Singleton
public class IceMachine {
    Foo foo;

    @Inject
    public IceMachine(Foo foo) {
        this.foo = foo;
    }
```

JSR 330 annotations → nothing new here

# TOOTHPICK FEATURES

## SCOPE

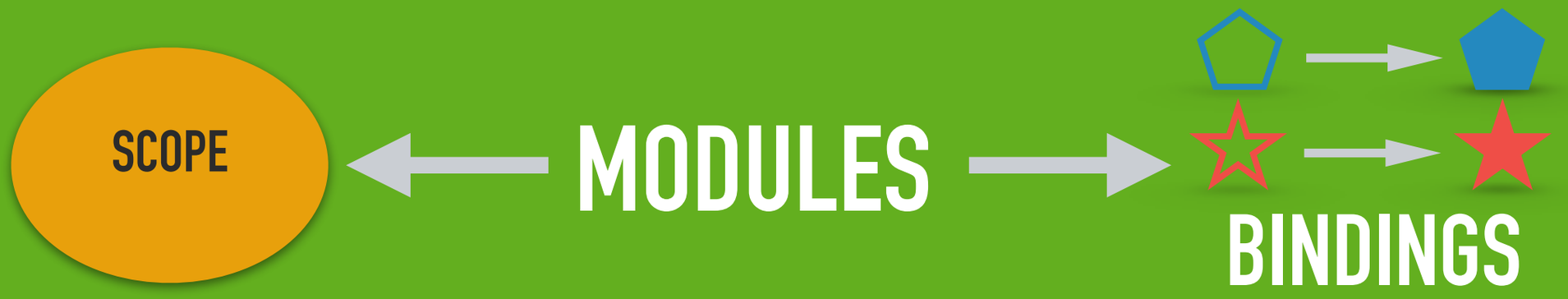
SCOPE

MAKE INJECTIONS

```
public class MyApplication extends Application {
  @Inject Machine machine;

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

    Scope appScope = Toothpick.openScope(this);
    appScope.installModules(…);
    Toothpick.inject(this, appScope);
  }
}
```

# TOOTHPICK FEATURES

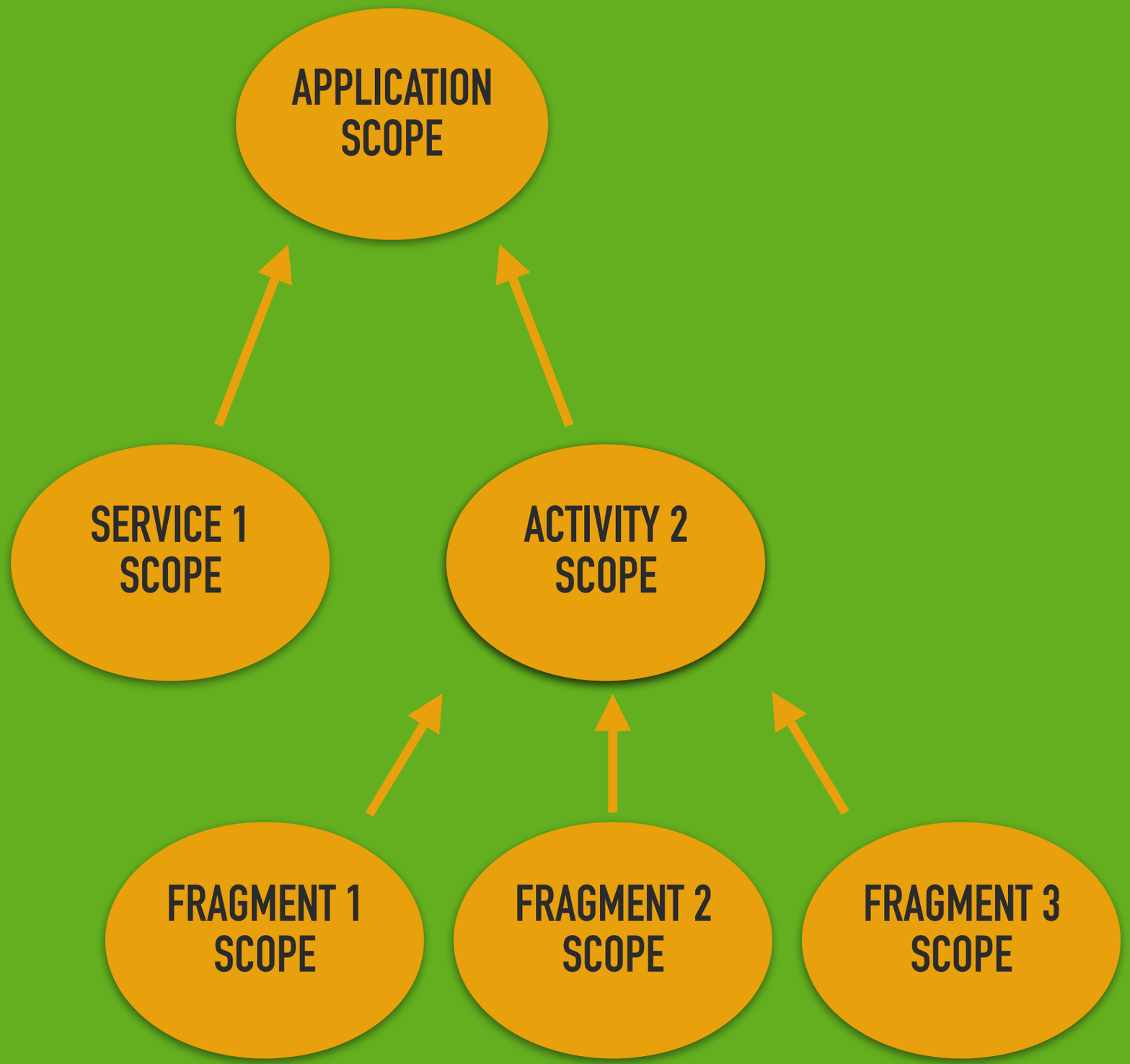
## SCOPE, MODULES & BINDINGS




```
public class MyApplication extends Application {
  @Inject Machine machine;

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

    Scope appScope = Toothpick.openScope(this);
    appScope.installModules(new Module() {{
      bind(Machine.class).to(IceMachine.class);
    }});
    Toothpick.inject(this, appScope);
  }
}
```

# TOOTHPICK FEATURES

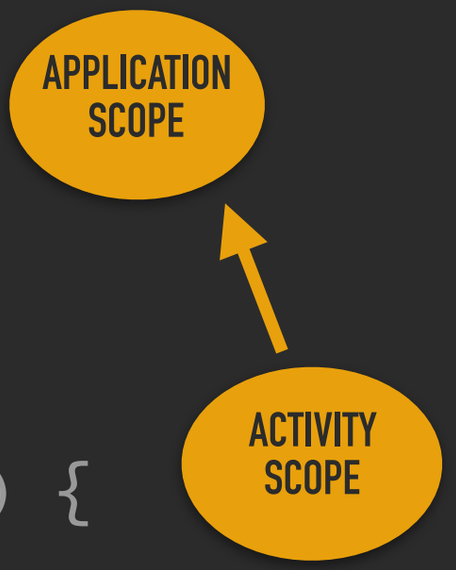
## SCOPE TREE



# TOOTHPICK FEATURES

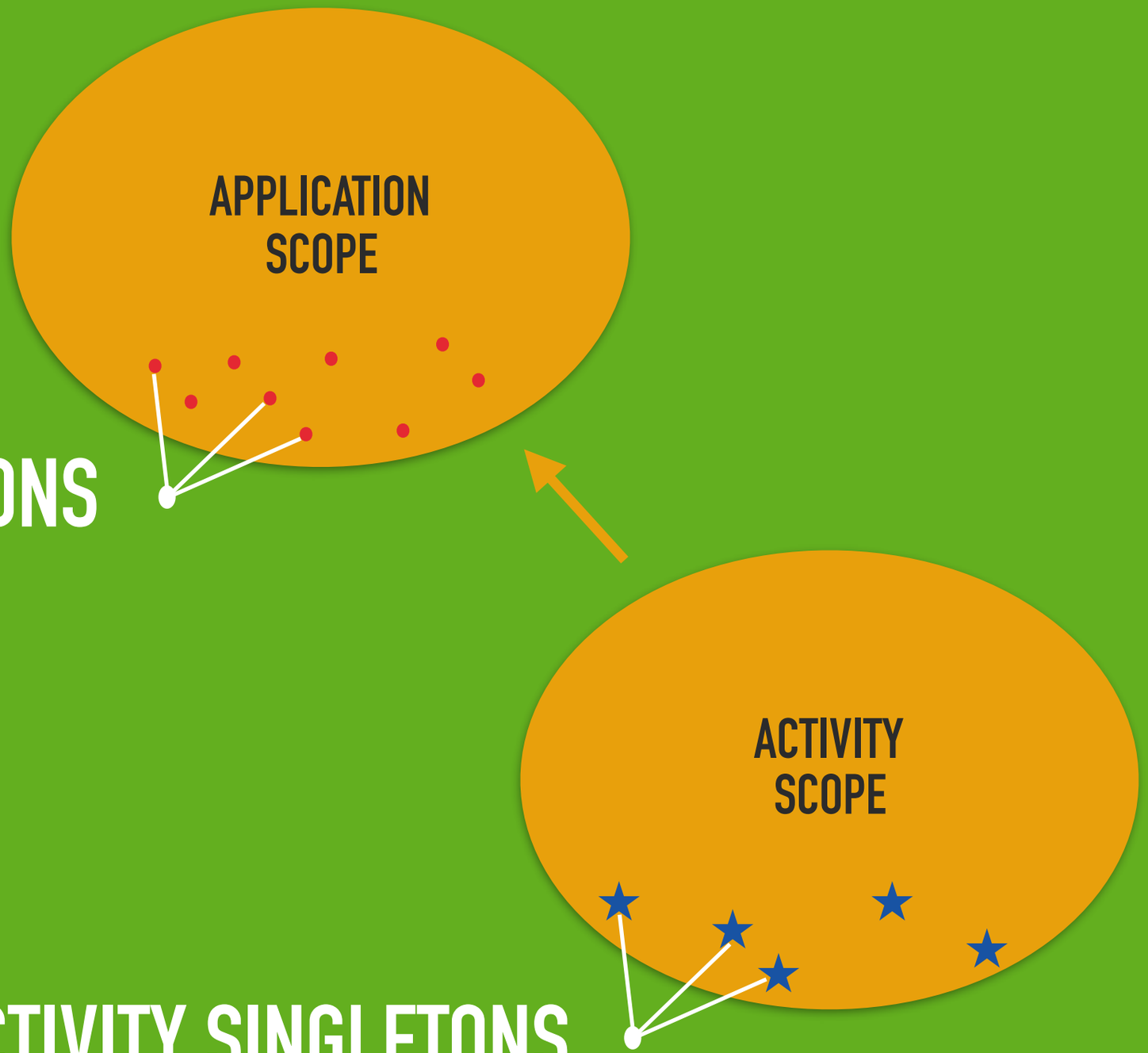
## SCOPE TREE

```
public class LemonActivity extends Activity {

  @Inject SmoothieMachine smoothieMachine;

 @Override
 protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    Scope scope = Toothpick.openScopes(getApplication(), this);
    scope.installModules(…);
    Toothpick.inject(this, scope);
 }
}
```



# TOOTHPICK FEATURES

## SCOPE SINGLETONS



# TOOTHPICK FEATURES

## SCOPE SINGLETONS

```
public class Module extends Module {
  public Module1() {
    bind(Machine.class).toInstance(new IceMachine());
  }
}
```

SINGLETONS CAN BE DEFINED IN MODULES

# TOOTHPICK FEATURES

## SCOPE SINGLETONS

```
@Singleton
public class IceMachine {
}
```


```
@ActivitySingleton
public class SmoothieMachine {
}
```

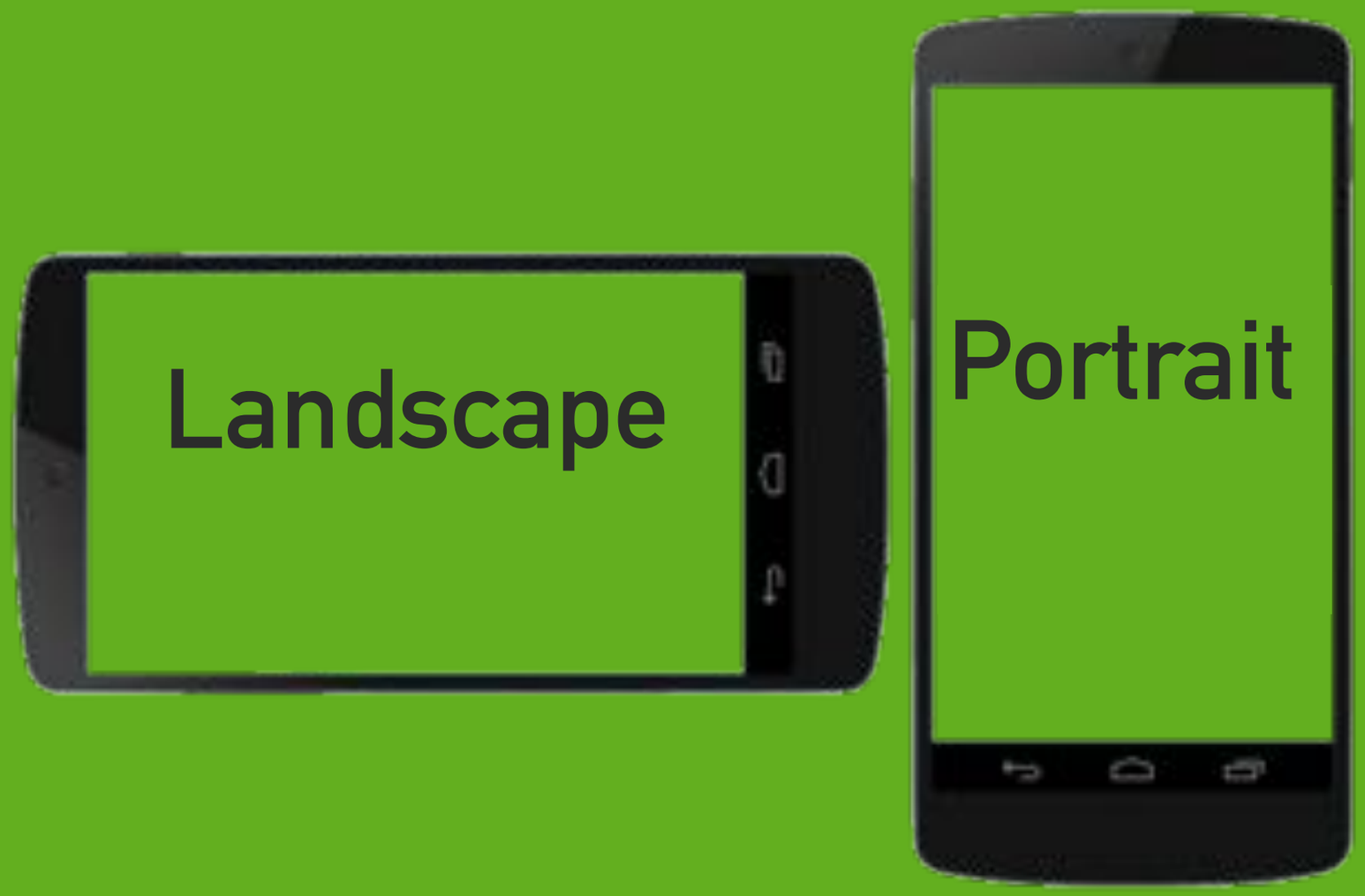

## OR USING ANNOTATIONS

# ADVANCED TOOTHPICK FEATURES

## MVP & STATE PRESERVATION

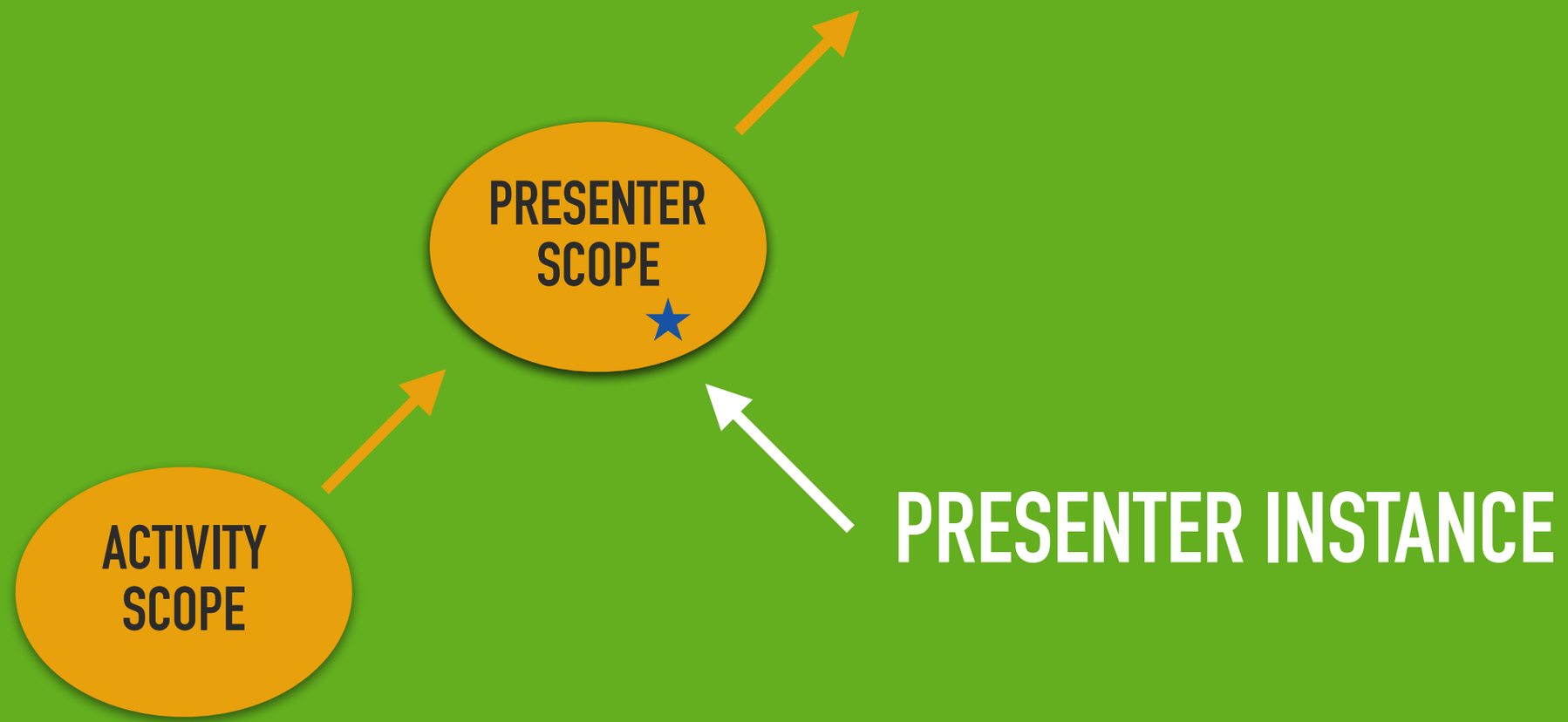
# ADVANCED TOOTHPICK FEATURES

## MVP & STATE PRESERVATION



# ADVANCED TOOTHPICK FEATURES

## MVP & STATE PRESERVATION



# ADVANCED TOOTHPICK FEATURES

## MVP & STATE PRESERVATION

```
public class MyActivity extends Activity {

 @Inject MyPresenter presenter;

 @Override
 protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   Scope scope = openScopes(getApplication(),
                            PresenterSingleton.class,
                            this);
   Toothpick.inject(this, scope);
 }
}
```

# ADVANCED TOOTHPICK FEATURES

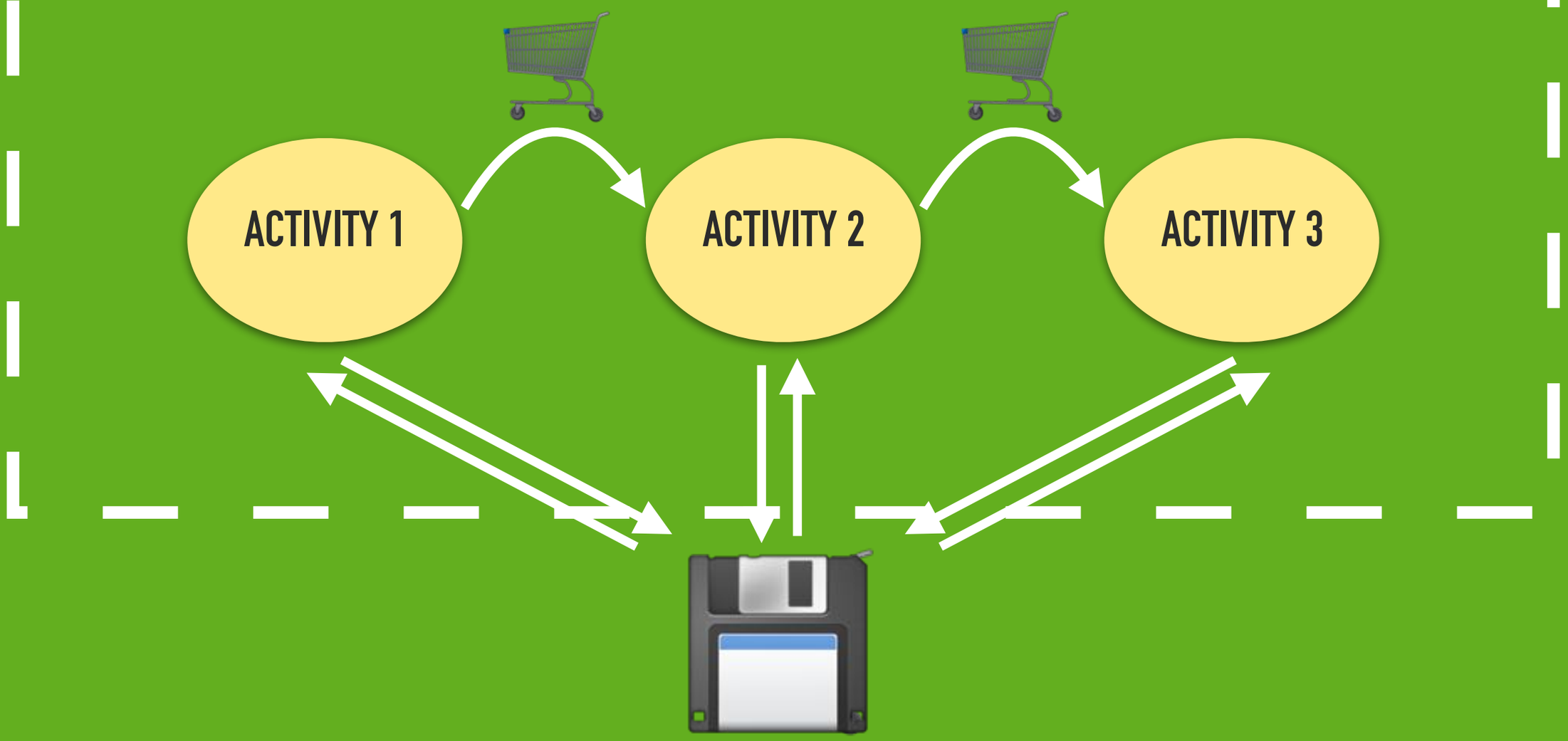
## MVP & STATE PRESERVATION


```
@PresenterSingleton
public class MyPresenter {
   String dealId;
   int quantity;
}
```

# ADVANCED TOOTHPICK FEATURES

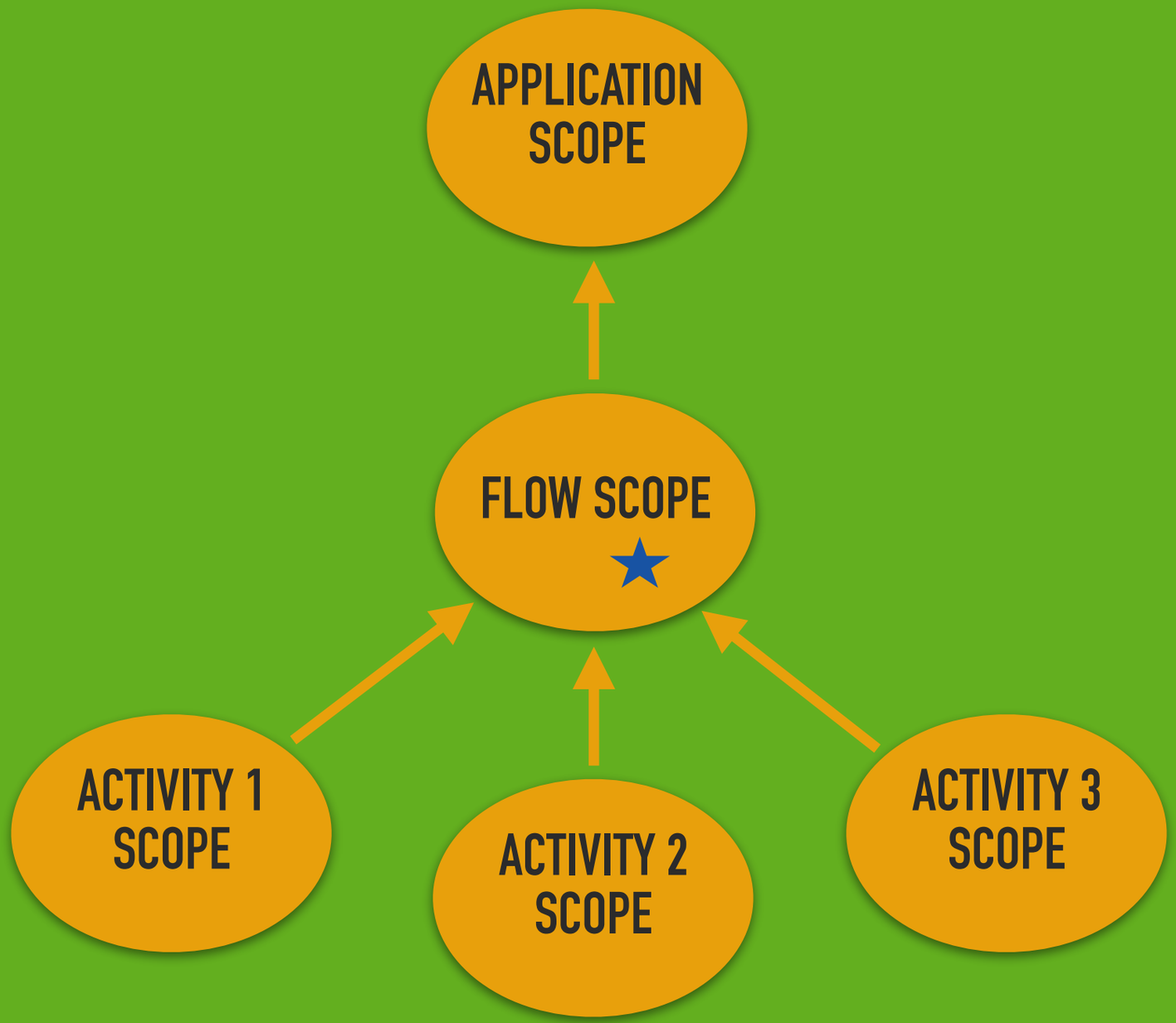
## MULTI ACTIVITY FLOWS

Purchase Flow



# ADVANCED TOOTHPICK FEATURES

## MULTI ACTIVITY FLOWS



# ADVANCED TOOTHPICK FEATURES

## MVP & STATE PRESERVATION

```
public class MyActivity extends Activity {

 @Inject ShoppingCart shoppingCart;

 @Override
 protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   Scope scope = openScopes(getApplication(),
                            FlowSingleton.class,
                            this);
   Toothpick.inject(this, scope);
 }
}
```

# ADVANCED TOOTHPICK FEATURES

## MULTI ACTIVITY FLOWS

FLOW SCOPE

```
@FlowSingleton
public class ShoppingCart {
  List<PurchaseItem> purchases…
}
```

# CONCLUSION

# TOOTHPICK

# A FRESH APPROACH TO DI

https://github.com/stephanenicolas/toothpick/

@D_Lemures

+stephane nicolas