Observer Pattern with Green Robot's Event Bus

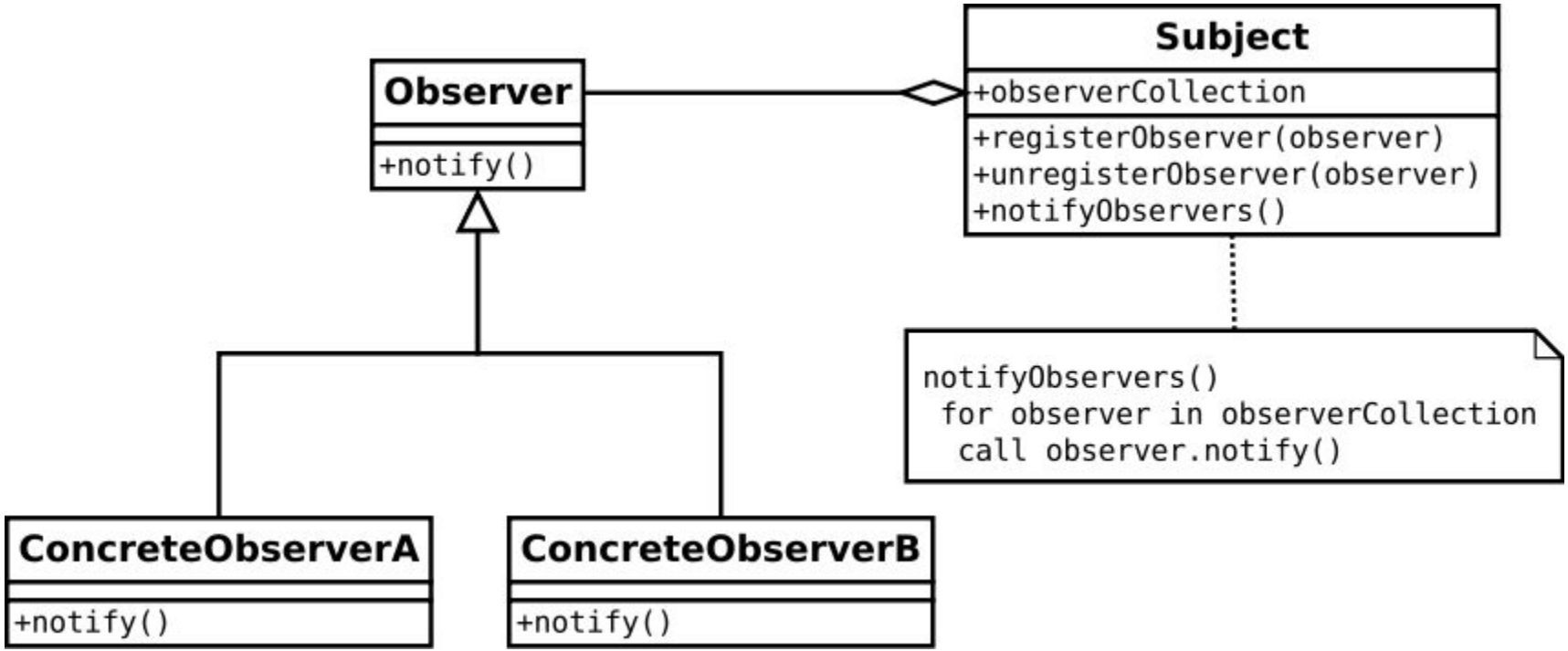
ANDROID

ElectronicArmory.com

The Observer Pattern

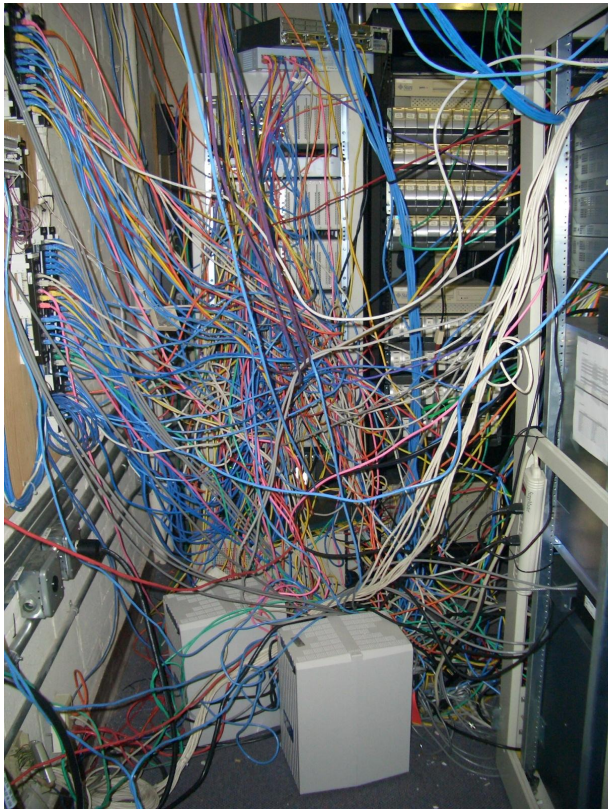
The **observer pattern** is a software design pattern in which an object, called the subject, maintains a list of its dependents, called observers, and notifies them automatically of any state changes, usually by calling one of their methods.

Observer Pattern



Low coupling, high cohesion

An example of high coupling.

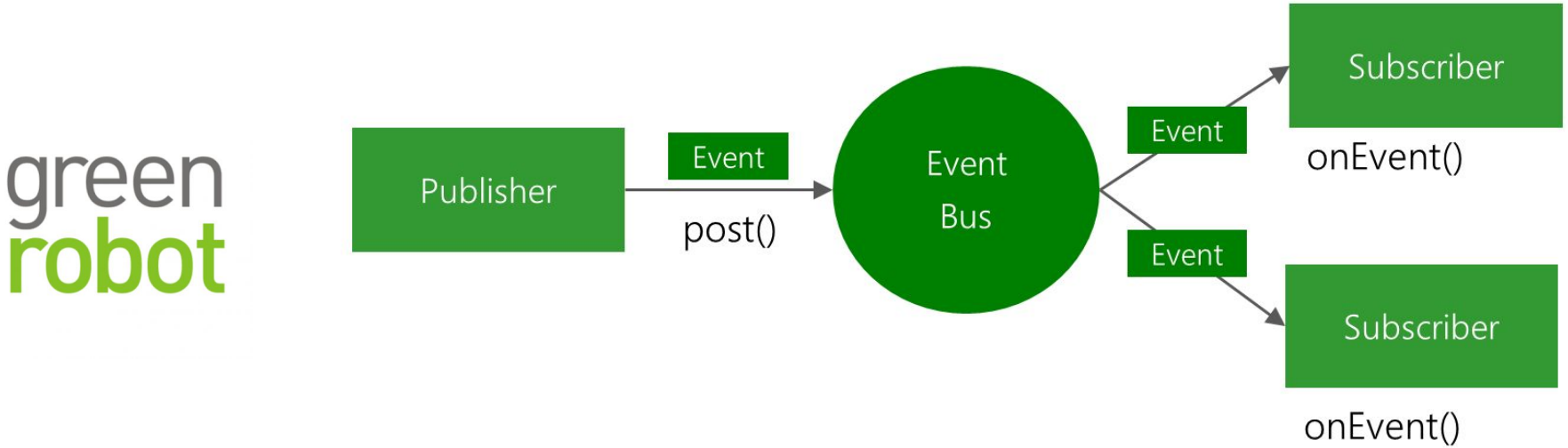


Low coupling, high cohesion

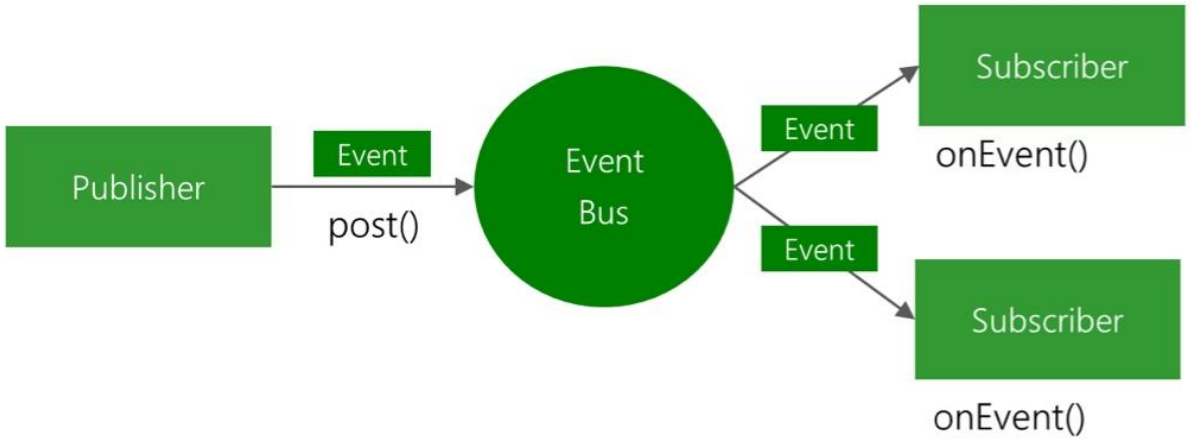
High cohesion refers to the degree to which functionality belongs to a class.

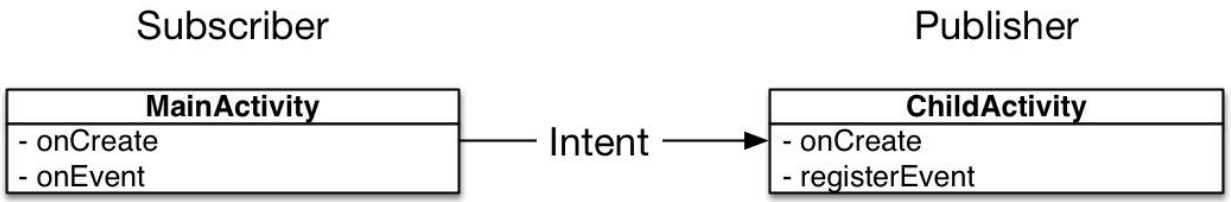
Event Bus

Event Bus architecture

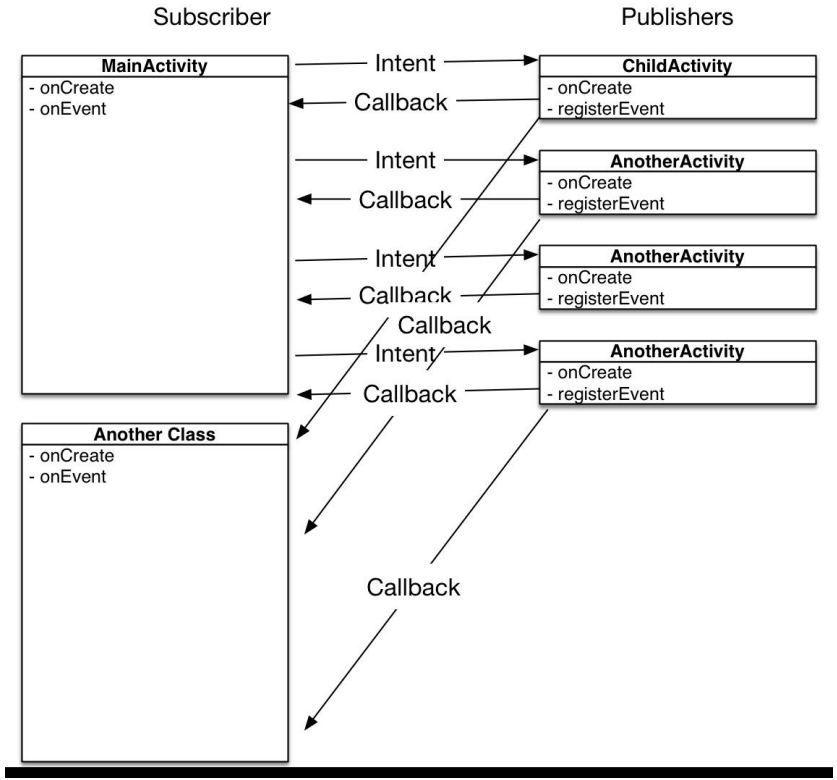


Our example application





How coupling can quickly get out of hand



1. Add the following to your Gradle file

```
implementation
'org.greenrobot:eventbus:3.1.1'
```

2. Define your message class

```
class CustomEvent {

   var payloadData:String? = null

}
```

3. Register a class to listen to events

```
EventBus.getDefault().register(this)
```

4. Subscribe to those events

```
public override fun onStart() {
    super.onStart()
    EventBus.getDefault().register(this)
}
```

5. Post those events

```
val newEvent:CustomEvent =
CustomEvent("ElectronicArmory.com")

EventBus.getDefault().post(newEvent)
```

6. Unregister (in onStop)

```
public override fun onStop() {
   super.onStop()
   EventBus.getDefault().unregister(this)
}
```