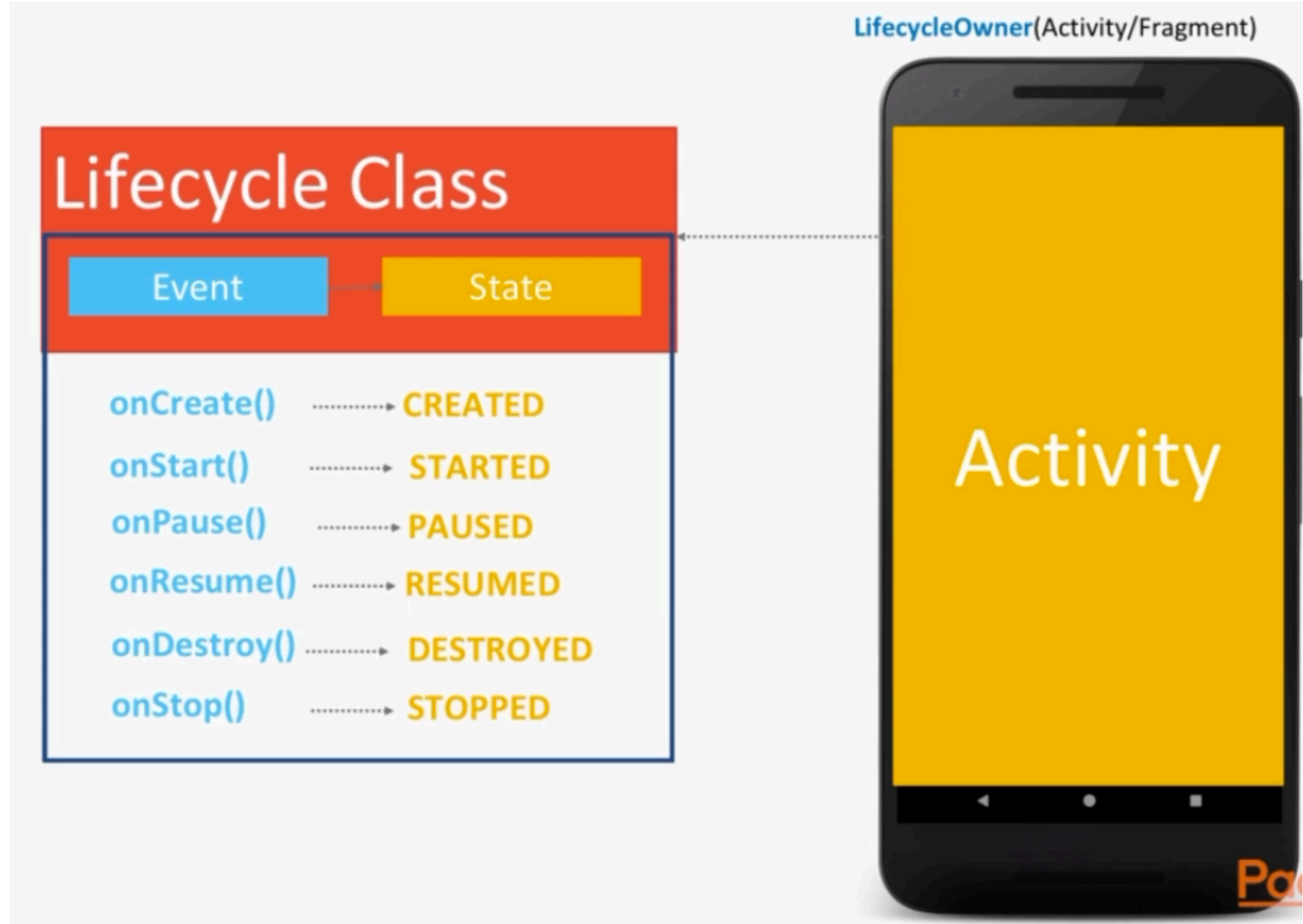
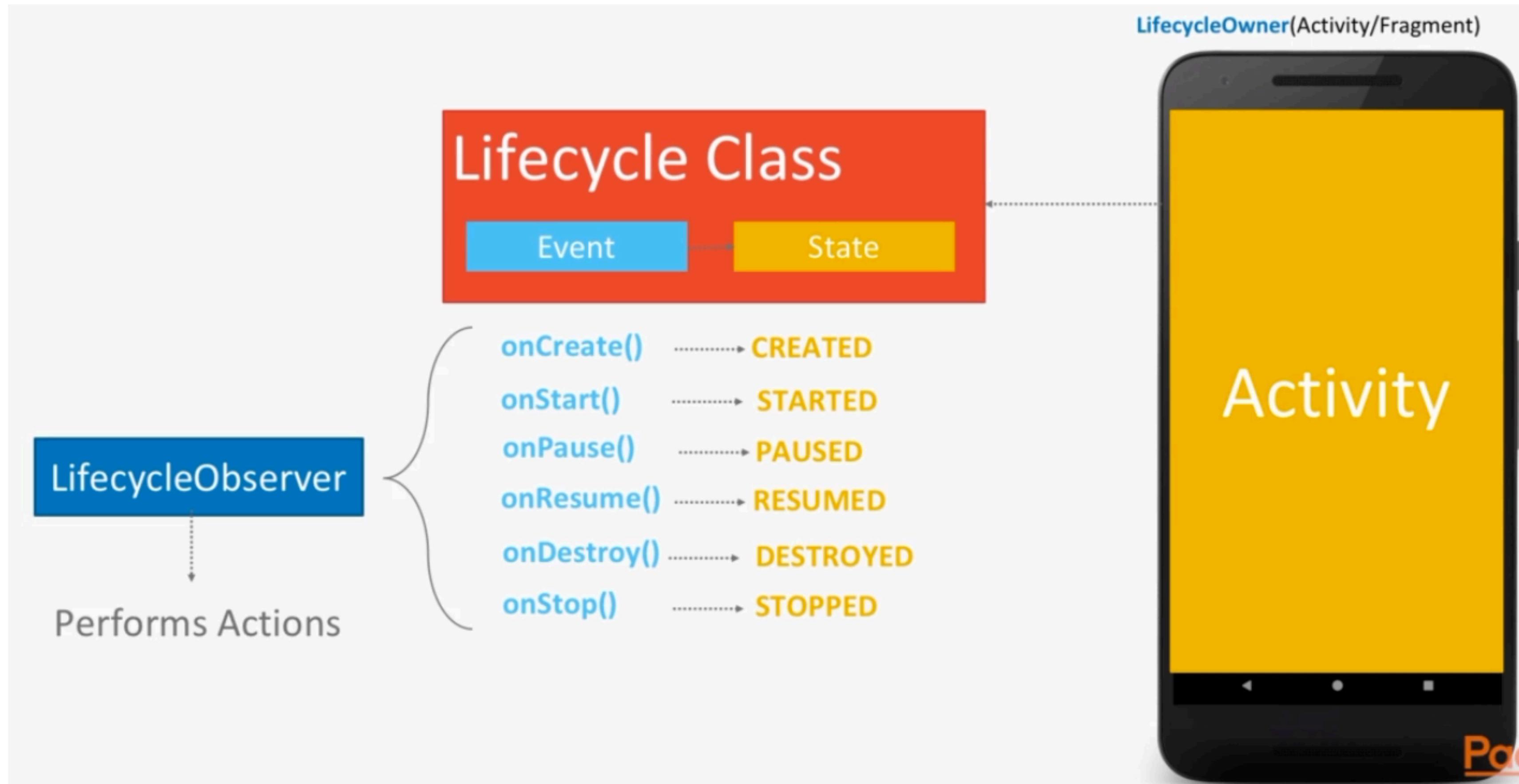


Lifecycle-Aware Components, ViewModel, LiveData

Jetpack Architecture





Documentation

- [OVERVIEW](#)
- [GUIDES](#)
- [REFERENCE](#)
- [SAMPLES](#)
- [DESIGN & QUALITY](#)

App Basics

- Introduction
- ▶ Build your first app
- App fundamentals
- ▶ App resources
- ▶ App manifest file
- ▶ App permissions

Devices

- ▶ Device compatibility
- ▶ Wear
- ▶ Android TV
- ▶ Android Auto
- ▶ Android Things
- ▶ Chrome OS devices

Core topics

- ▶ Activities
- ▼ Architecture Components

Handling Lifecycles with Lifecycle-Aware Components



Part of [Android Jetpack](#).

Lifecycle-aware components perform actions in response to a change in the lifecycle status of another component, such as activities and fragments. These components help you produce better-organized, and often lighter-weight code, that is easier to maintain.

A common pattern is to implement the actions of the dependent components in the lifecycle methods of activities and fragments. However, this pattern leads to a poor organization of the code and to the proliferation of errors. By using lifecycle-aware components, you can move the code of dependent components out of the lifecycle methods and into the components themselves.

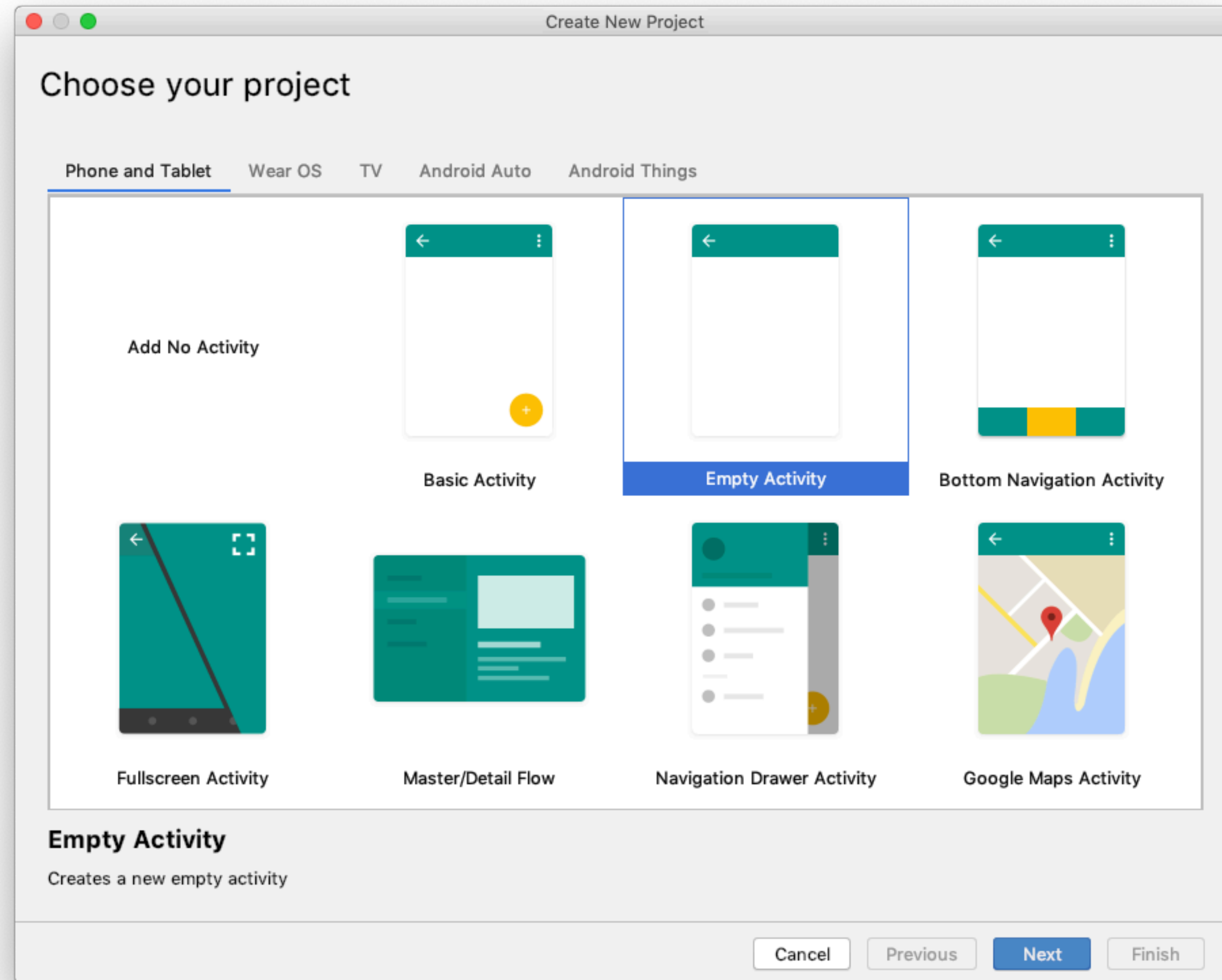
The `android.arch.lifecycle` package provides classes and interfaces that let you build *lifecycle-aware* components —which are components that can automatically adjust their behavior based on the current lifecycle state of an activity or fragment.

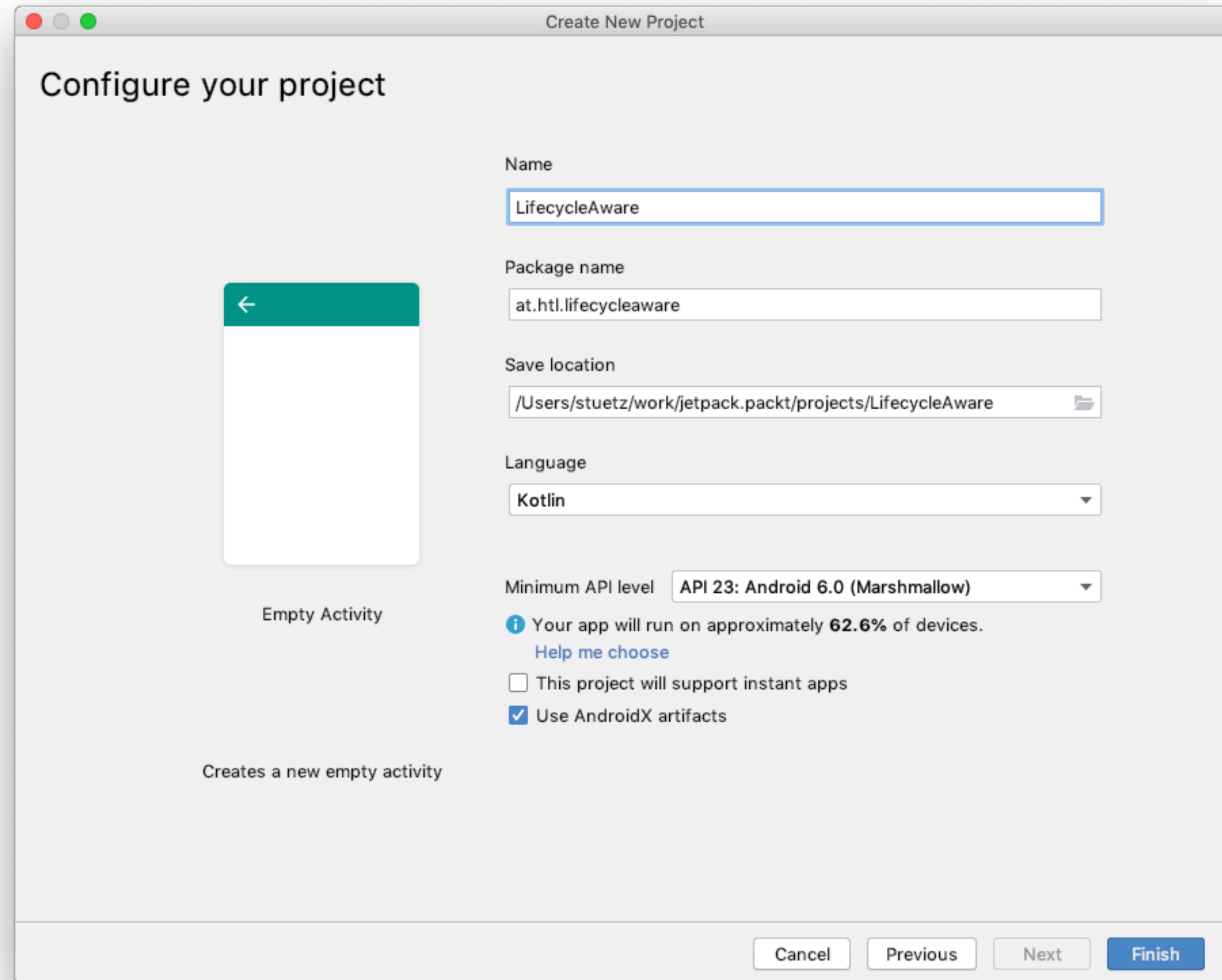
★ **Note:** To import `android.arch.lifecycle` into your Android project, see the instructions for declaring dependencies in the [Lifecycle release notes](#).

Contents

- Lifecycle
- LifecycleOwner
 - Implementing a custom LifecycleOwner
- Best practices for lifecycle-aware components
- Use cases for lifecycle-aware components
- Handling on stop events
- Additional resources

Lifecycle-Aware Components





The screenshot shows an IDE window for an Android project named "LifecycleAware". The main editor displays the Kotlin code for MainActivity.kt. The code is as follows:

```
1 package at.htl.lifecycleaware
2
3 import androidx.appcompat.app.AppCompatActivity
4 import android.os.Bundle
5
6 class MainActivity : AppCompatActivity() {
7
8     override fun onCreate(savedInstanceState: Bundle?) {
9         super.onCreate(savedInstanceState)
10        setContentView(R.layout.activity_main)
11    }
12 }
13
```

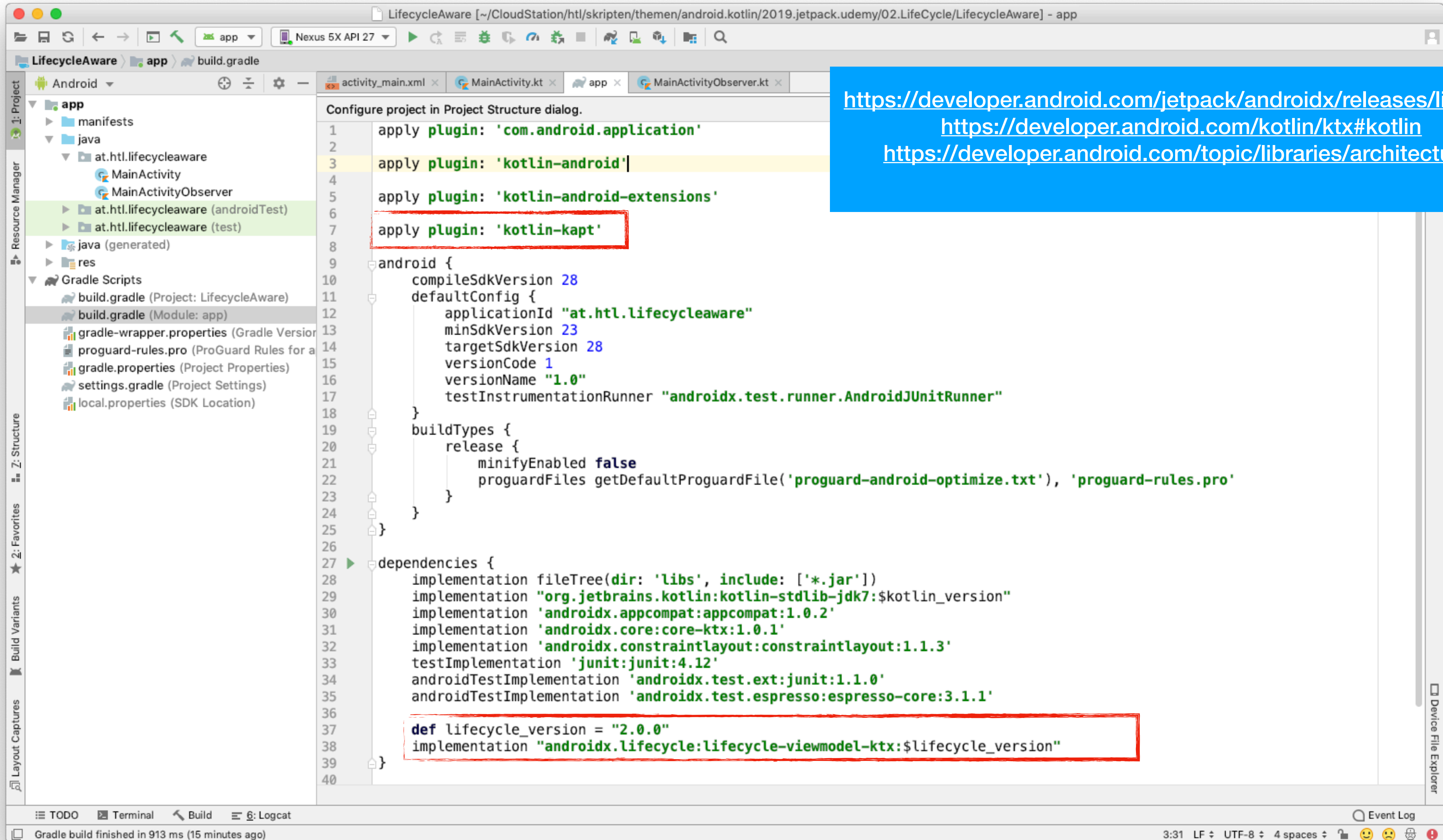
The left sidebar shows the project structure with the following folders:

- app
 - manifests
 - java
 - at.htl.lifecycleaware
 - MainActivity
 - at.htl.lifecycleaware (androidTest)
 - at.htl.lifecycleaware (test)
 - java (generated)
 - res
 - Gradle Scripts

The bottom panel shows the Build Output window with the following log:

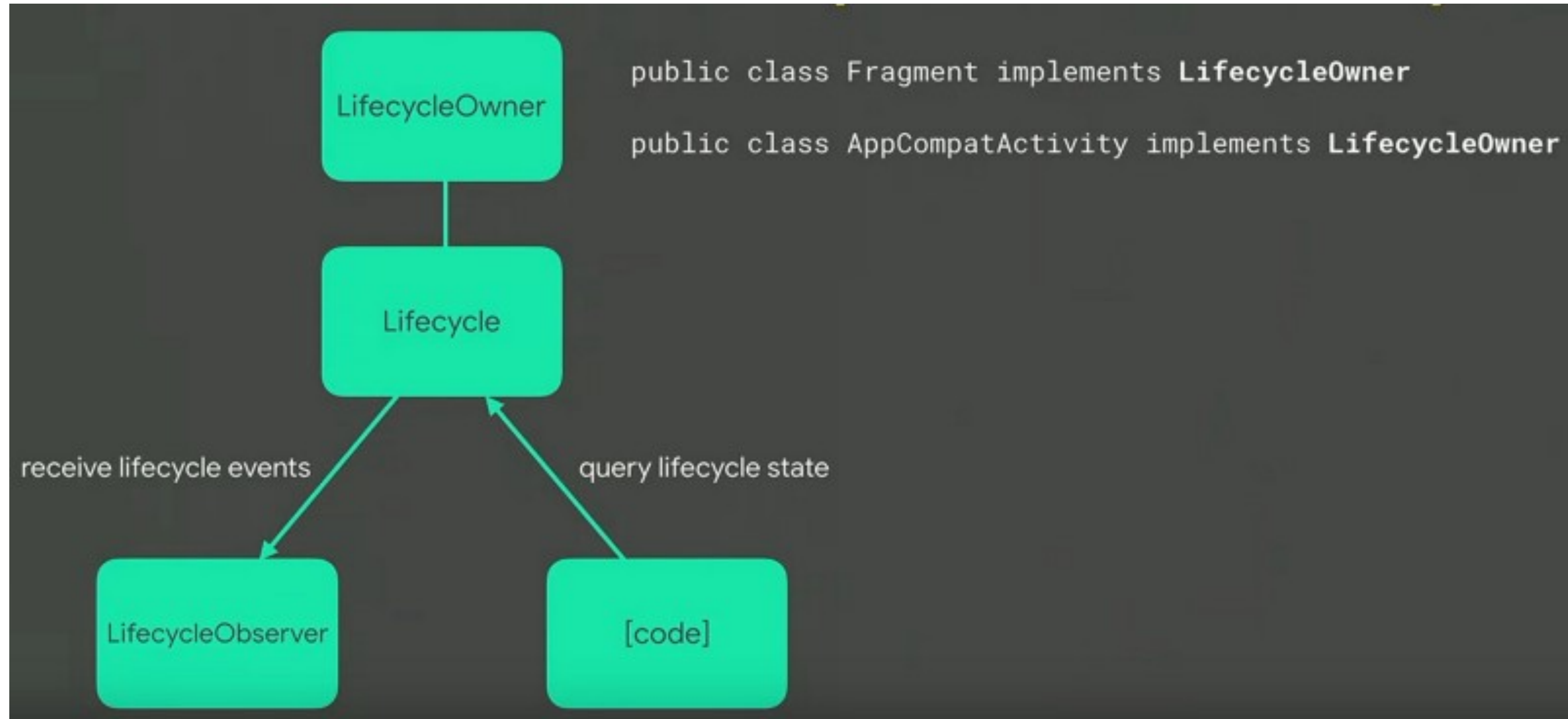
```
Build: Build Output x Sync x
Build: completed successfully at 2019-02-17 20:33 841 ms
  Run build /Users/stuetz/CloudStation/htl/skripten/themen/android.kotlin/2019.jetpack.udemy/02.LifeCycle/LifecycleAware 529 ms
    Load build 5 ms
    Configure build 293 ms
    Calculate task graph 61 ms
    Run tasks 169 ms
```

The status bar at the bottom indicates "Gradle build finished in 856 ms (moments ago)" and "13:1 LF UTF-8 4 spaces".

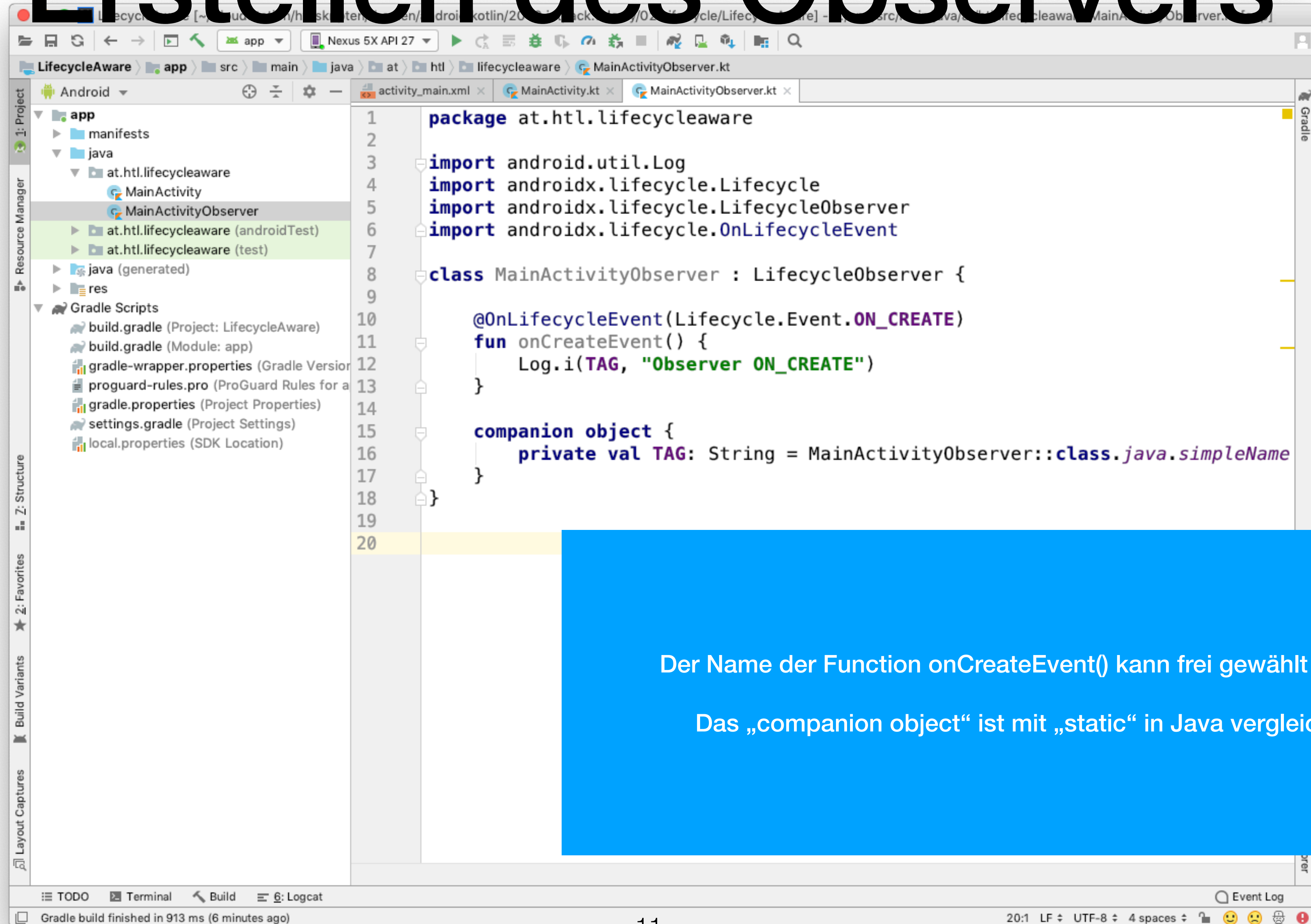


<https://developer.android.com/jetpack/androidx/releases/lifecycle>
<https://developer.android.com/kotlin/ktx#kotlin>
<https://developer.android.com/topic/libraries/architecture/>

<https://developer.android.com/jetpack/androidx/releases/lifecycle>



Erstellen des Observers



The screenshot shows an IDE window for an Android project named "LifecycleAware". The project structure on the left includes a package "at.htl.lifecycleaware" with classes "MainActivity" and "MainActivityObserver". The "MainActivityObserver.kt" file is open in the editor, showing the following code:

```
1 package at.htl.lifecycleaware
2
3 import android.util.Log
4 import androidx.lifecycle.Lifecycle
5 import androidx.lifecycle.LifecycleObserver
6 import androidx.lifecycle.OnLifecycleEvent
7
8 class MainActivityObserver : LifecycleObserver {
9
10     @OnLifecycleEvent(Lifecycle.Event.ON_CREATE)
11     fun onCreateEvent() {
12         Log.i(TAG, "Observer ON_CREATE")
13     }
14
15     companion object {
16         private val TAG: String = MainActivityObserver::class.java.simpleName
17     }
18 }
19
20
```

A blue callout box contains the following text:

Der Name der Function onCreateEvent() kann frei gewählt werden.

Das „companion object“ ist mit „static“ in Java vergleichbar

The IDE interface includes a Project Manager, Resource Manager, and a bottom toolbar with options like TODO, Terminal, Build, Logcat, and Event Log. The status bar at the bottom indicates "Gradle build finished in 913 ms (6 minutes ago)" and "20:1 LF UTF-8 4 spaces".

Registrieren des Observers beim Owner

Die Activity ist der Lifecycle-Owner

```
1 package at.htl.lifecycleaware
2
3 import androidx.appcompat.app.AppCompatActivity
4 import android.os.Bundle
5 import android.util.Log
6
7 class MainActivity : AppCompatActivity() {
8
9     override fun onCreate(savedInstanceState: Bundle?) {
10         super.onCreate(savedInstanceState)
11         setContentView(R.layout.activity_main)
12
13         Log.i(TAG, "attached Observer to Owner - onCreate")
14         lifecycle.addObserver(MainActivityObserver())
15     }
16
17     companion object {
18         private val TAG: String = MainActivity::class.java.simpleName
19     }
20 }
21
```

Log.i(TAG, "attached Observer to Owner - onCreate")
lifecycle.addObserver(MainActivityObserver())

companion object {
private val TAG: String = MainActivity::class.java.simpleName
}

The screenshot shows an IDE window for a project named "LifecycleAware". The main editor displays the following Kotlin code for MainActivity.kt:

```
1 import ...
2
3
4
5
6
7 class MainActivity : AppCompatActivity() {
8
9     override fun onCreate(savedInstanceState: Bundle?) {
10         super.onCreate(savedInstanceState)
11         setContentView(R.layout.activity_main)
12
13         Log.i(TAG, "attached Observer to Owner - onCreate")
14         lifecycle.addObserver(MainActivityObserver())
15     }
16
17     companion object {
18         private val TAG: String = MainActivity::class.java.simpleName
19     }
20 }
```

The Logcat window at the bottom shows the following output, with a search filter "Main" applied:

```
2019-02-17 21:14:17.612 3968-3968/at.htl.lifecycleaware I/zygote: at void at.htl.lifecycleaware.MainActivity.onCreate(android.os.Bundle) (MainActivity.kt:11)
2019-02-17 21:14:17.612 3968-3968/at.htl.lifecycleaware I/zygote: at void at.htl.lifecycleaware.MainActivity.onCreate(android.os.Bundle) (MainActivity.kt:11)
2019-02-17 21:14:17.614 3968-3968/at.htl.lifecycleaware I/zygote: at void at.htl.lifecycleaware.MainActivity.onCreate(android.os.Bundle) (MainActivity.kt:11)
2019-02-17 21:14:17.615 3968-3968/at.htl.lifecycleaware I/zygote: at void at.htl.lifecycleaware.MainActivity.onCreate(android.os.Bundle) (MainActivity.kt:11)
2019-02-17 21:14:17.616 3968-3968/at.htl.lifecycleaware I/zygote: at void at.htl.lifecycleaware.MainActivity.onCreate(android.os.Bundle) (MainActivity.kt:11)
2019-02-17 21:14:17.618 3968-3968/at.htl.lifecycleaware I/zygote: at void at.htl.lifecycleaware.MainActivity.onCreate(android.os.Bundle) (MainActivity.kt:11)
2019-02-17 21:14:17.688 3968-3968/at.htl.lifecycleaware I/MainActivity: attached Observer to Owner - onCreate
2019-02-17 21:14:17.707 3968-3968/at.htl.lifecycleaware I/MainActivityObserver: Observer ON_CREATE
```

Annotations in the image:

- A blue box with the text "Filtern Sie die Ausgaben mit „Main“" points to the search filter "Main" in the Logcat window.
- Another blue box with the text "Hier sieht man das Ergebnis" points to the log output line: "2019-02-17 21:14:17.688 3968-3968/at.htl.lifecycleaware I/MainActivity: attached Observer to Owner - onCreate".

On the right, a smartphone displays the app's interface with a green header "LifecycleAware" and the text "Hello World!" on a white background.

The screenshot shows an IDE window for a project named 'LifecycleAware'. The main editor displays the Kotlin code for MainActivity.kt. The code is as follows:

```
1 package at.htl.lifecycleaware
2
3 import ...
4
5
6
7 class MainActivity : AppCompatActivity() {
8
9     override fun onCreate(savedInstanceState: Bundle?) {...}
10
11
12
13
14
15
16
17     override fun onStart() {
18         super.onStart()
19         Log.i(TAG, "Owner onStart")
20     }
21
22
23     override fun onPause() {
24         super.onPause()
25         Log.i(TAG, "Owner onPause")
26     }
27
28
29     override fun onResume() {
30         super.onResume()
31         Log.i(TAG, "Owner onResume")
32     }
33
34
35
36
37     override fun onDestroy() {
38         super.onDestroy()
39         Log.i(TAG, "Owner onDestroy")
40     }
41
42
43     override fun onStop() {
44         super.onStop()
45         Log.i(TAG, "Owner onStop")
46     }
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

An 'Override Members' dialog box is open, showing a search for 'onStart'. The list of methods includes 'onStart(): Unit', which is highlighted. The dialog also has a 'Copy JavaDoc' checkbox and buttons for 'Select None', 'Cancel', and 'OK'.

Annotations on the image include:

- A red box highlights the `onStart()` method in the code.
- A green box contains the text: **Override Methods via ^O (Ctrl+O for Win/Linux)**
- A blue box contains the text: **Mit Strg-O kann man das „Override Members“-Fenster öffnen.**
Nun gibt man zB „onStart“ ein.
Anschließend navigiert man mit der Pfeiltaste ↓ bis man die korrekte Methode findet.

```
1 /
2
3
4
5
6
7
8 class MainActivityObserver : LifecycleObserver {
9
10     @OnLifecycleEvent(Lifecycle.Event.ON_CREATE)
11     fun onCreateEvent() {
12         Log.i(TAG, "Observer ON_CREATE")
13     }
14
15     @OnLifecycleEvent(Lifecycle.Event.ON_START)
16     fun onStartEvent() {
17         Log.i(TAG, "Observer ON_START")
18     }
19
20     @OnLifecycleEvent(Lifecycle.Event.ON_RESUME)
21     fun onResumeEvent() {
22         Log.i(TAG, "Observer ON_RESUME")
23     }
24
25     @OnLifecycleEvent(Lifecycle.Event.ON_PAUSE)
26     fun onPauseEvent() {
27         Log.i(TAG, "Observer ON_PAUSE")
28     }
29
30     @OnLifecycleEvent(Lifecycle.Event.ON_STOP)
31     fun onStopEvent() {
32         Log.i(TAG, "Observer ON_STOP")
33     }
34
35     @OnLifecycleEvent(Lifecycle.Event.ON_DESTROY)
36     fun onDestroyEvent() {
37         Log.i(TAG, "Observer ON_DESTROY")
38     }
39
40     companion object {
```

The screenshot shows the Logcat window in an IDE. The selected application is 'Emulator Nexus_5X_API_27 And' with package 'at.htl.lifecycleaware' (PID 4664). The log level is set to 'Verbose' and the filter is 'Main'. The log entries are as follows:

```
2019-02-17 21:44:53.652 4664-4664/at.htl.lifecycleaware I/zygote: at void at.htl.lifecycleaware.MainActivity.onCreate(android.os.Bundle) (MainActivity.kt:11)
2019-02-17 21:44:53.653 4664-4664/at.htl.lifecycleaware I/zygote: at void at.htl.lifecycleaware.MainActivity.onCreate(android.os.Bundle) (MainActivity.kt:11)
2019-02-17 21:44:53.653 4664-4664/at.htl.lifecycleaware I/zygote: at void at.htl.lifecycleaware.MainActivity.onCreate(android.os.Bundle) (MainActivity.kt:11)
2019-02-17 21:44:53.653 4664-4664/at.htl.lifecycleaware I/zygote: at void at.htl.lifecycleaware.MainActivity.onCreate(android.os.Bundle) (MainActivity.kt:11)
2019-02-17 21:44:53.654 4664-4664/at.htl.lifecycleaware I/zygote: at void at.htl.lifecycleaware.MainActivity.onCreate(android.os.Bundle) (MainActivity.kt:11)
2019-02-17 21:44:53.654 4664-4664/at.htl.lifecycleaware I/zygote: at void at.htl.lifecycleaware.MainActivity.onCreate(android.os.Bundle) (MainActivity.kt:11)
2019-02-17 21:44:53.702 4664-4664/at.htl.lifecycleaware I/MainActivity: attached Observer to Owner - onCreate
2019-02-17 21:44:53.706 4664-4664/at.htl.lifecycleaware I/MainActivityObserver: Observer ON_CREATE
2019-02-17 21:44:53.710 4664-4664/at.htl.lifecycleaware I/MainActivity: Owner onStart
2019-02-17 21:44:53.710 4664-4664/at.htl.lifecycleaware I/MainActivityObserver: Observer ON_START
2019-02-17 21:44:53.713 4664-4664/at.htl.lifecycleaware I/MainActivity: Owner onResume
2019-02-17 21:44:53.713 4664-4664/at.htl.lifecycleaware I/MainActivityObserver: Observer ON_RESUME
```

A blue callout box contains the text: "Die App wird initialisiert. Die Callback-Methoden werden das erste Mal aufgerufen".

A green notification at the bottom of the IDE says "Install successful".



The screenshot shows the Android Studio interface. The top toolbar includes icons for running, debugging, and other development actions. The breadcrumb navigation shows the path: LifecycleAware > app > src > main > java > at > htl > lifecycleaware > MainActivityObserver.kt. The Logcat window is open, displaying logs for the application 'at.htl.lifecycleaware' (PID 4664) on the 'Emulator Nexus_5X_API_27 And' device. The logs show the lifecycle of the MainActivity, including onCreate, onStart, onResume, onPause, onStop, onDestroy, and onCreate again. A red box highlights the log entries from 21:45:32.178 to 21:45:32.236, which correspond to the activity being destroyed and recreated. A blue callout box with white text states: "Beim Drehen des Smartphones wird die Activity zerstört und neu erstellt". At the bottom of the IDE, a green notification bubble says "Install successful". The smartphone emulator at the bottom right shows the app's UI with a green header and the text "Hello World".

```
2019-02-17 21:44:53.652 4664-4664/at.htl.lifecycleaware I/zygote: at void at.htl.lifecycleaware.MainActivity.onCreate(android.os.Bundle) (MainActivity.kt:11)
2019-02-17 21:44:53.653 4664-4664/at.htl.lifecycleaware I/zygote: at void at.htl.lifecycleaware.MainActivity.onCreate(android.os.Bundle) (MainActivity.kt:11)
2019-02-17 21:44:53.653 4664-4664/at.htl.lifecycleaware I/zygote: at void at.htl.lifecycleaware.MainActivity.onCreate(android.os.Bundle) (MainActivity.kt:11)
2019-02-17 21:44:53.653 4664-4664/at.htl.lifecycleaware I/zygote: at void at.htl.lifecycleaware.MainActivity.onCreate(android.os.Bundle) (MainActivity.kt:11)
2019-02-17 21:44:53.654 4664-4664/at.htl.lifecycleaware I/zygote: at void at.htl.lifecycleaware.MainActivity.onCreate(android.os.Bundle) (MainActivity.kt:11)
2019-02-17 21:44:53.654 4664-4664/at.htl.lifecycleaware I/zygote: at void at.htl.lifecycleaware.MainActivity.onCreate(android.os.Bundle) (MainActivity.kt:11)
2019-02-17 21:44:53.702 4664-4664/at.htl.lifecycleaware I/MainActivity: attached Observer to Owner - onCreate
2019-02-17 21:44:53.706 4664-4664/at.htl.lifecycleaware I/MainActivityObserver: Observer ON_CREATE
2019-02-17 21:44:53.710 4664-4664/at.htl.lifecycleaware I/MainActivity: Owner onStart
2019-02-17 21:44:53.710 4664-4664/at.htl.lifecycleaware I/MainActivityObserver: Observer ON_START
2019-02-17 21:44:53.713 4664-4664/at.htl.lifecycleaware I/MainActivity: Owner onResume
2019-02-17 21:44:53.713 4664-4664/at.htl.lifecycleaware I/MainActivityObserver: Observer ON_RESUME
2019-02-17 21:45:32.178 4664-4664/at.htl.lifecycleaware I/MainActivityObserver: Observer ON_PAUSE
2019-02-17 21:45:32.178 4664-4664/at.htl.lifecycleaware I/MainActivity: Owner onPause
2019-02-17 21:45:32.178 4664-4664/at.htl.lifecycleaware I/MainActivityObserver: Observer ON_STOP
2019-02-17 21:45:32.181 4664-4664/at.htl.lifecycleaware I/MainActivity: Owner onStop
2019-02-17 21:45:32.182 4664-4664/at.htl.lifecycleaware I/MainActivityObserver: Observer ON_DESTROY
2019-02-17 21:45:32.182 4664-4664/at.htl.lifecycleaware I/MainActivity: Owner onDestroy
2019-02-17 21:45:32.231 4664-4664/at.htl.lifecycleaware I/MainActivity: attached Observer to Owner - onCreate
2019-02-17 21:45:32.231 4664-4664/at.htl.lifecycleaware I/MainActivityObserver: Observer ON_CREATE
2019-02-17 21:45:32.232 4664-4664/at.htl.lifecycleaware I/MainActivity: Owner onStart
2019-02-17 21:45:32.232 4664-4664/at.htl.lifecycleaware I/MainActivityObserver: Observer ON_START
2019-02-17 21:45:32.236 4664-4664/at.htl.lifecycleaware I/MainActivity: Owner onResume
2019-02-17 21:45:32.236 4664-4664/at.htl.lifecycleaware I/MainActivityObserver: Observer ON_RESUME
```

LifecycleAware [~/CloudStation/htl/skripten/themen/android.kotlin/2019.jetpack.udemy/02.LifeCycle/LifecycleAware] - .../app/src/main/java/at/htl/lifecycleaware/MainActivityObserver.kt [app]

app

Logcat

Emulator Nexus_5X_API_27 And at.htl.lifecycleaware (4664) Verbose Main

```

2019-02-17 21:44:53.652 4664-4664/at.htl.lifecycleaware I/zygote: at void at.htl.lifecycleaware.MainActivity.onCreate(android.os.Bundle) (MainActivity.kt:11)
2019-02-17 21:44:53.653 4664-4664/at.htl.lifecycleaware I/zygote: at void at.htl.lifecycleaware.MainActivity.onCreate(android.os.Bundle) (MainActivity.kt:11)
2019-02-17 21:44:53.653 4664-4664/at.htl.lifecycleaware I/zygote: at void at.htl.lifecycleaware.MainActivity.onCreate(android.os.Bundle) (MainActivity.kt:11)
2019-02-17 21:44:53.653 4664-4664/at.htl.lifecycleaware I/zygote: at void at.htl.lifecycleaware.MainActivity.onCreate(android.os.Bundle) (MainActivity.kt:11)
2019-02-17 21:44:53.654 4664-4664/at.htl.lifecycleaware I/zygote: at void at.htl.lifecycleaware.MainActivity.onCreate(android.os.Bundle) (MainActivity.kt:11)
2019-02-17 21:44:53.654 4664-4664/at.htl.lifecycleaware I/zygote: at void at.htl.lifecycleaware.MainActivity.onCreate(android.os.Bundle) (MainActivity.kt:11)
2019-02-17 21:44:53.702 4664-4664/at.htl.lifecycleaware I/MainActivity: attached Observer to Owner - onCreate
2019-02-17 21:44:53.706 4664-4664/at.htl.lifecycleaware I/MainActivityObserver: Observer ON_CREATE
2019-02-17 21:44:53.710 4664-4664/at.htl.lifecycleaware I/MainActivity: Owner onStart
2019-02-17 21:44:53.710 4664-4664/at.htl.lifecycleaware I/MainActivityObserver: Observer ON_START
2019-02-17 21:44:53.713 4664-4664/at.htl.lifecycleaware I/MainActivity: Owner onResume
2019-02-17 21:44:53.713 4664-4664/at.htl.lifecycleaware I/MainActivityObserver: Observer ON_RESUME
2019-02-17 21:45:32.178 4664-4664/at.htl.lifecycleaware I/MainActivityObserver: Observer ON_PAUSE
2019-02-17 21:45:32.178 4664-4664/at.htl.lifecycleaware I/MainActivity: Owner onPause
2019-02-17 21:45:32.178 4664-4664/at.htl.lifecycleaware I/MainActivityObserver: Observer ON_STOP
2019-02-17 21:45:32.181 4664-4664/at.htl.lifecycleaware I/MainActivity: Owner onStop
2019-02-17 21:45:32.182 4664-4664/at.htl.lifecycleaware I/MainActivityObserver: Observer ON_DESTROY
2019-02-17 21:45:32.182 4664-4664/at.htl.lifecycleaware I/MainActivity: Owner onDestroy
2019-02-17 21:45:32.231 4664-4664/at.htl.lifecycleaware I/MainActivity: attached Observer to Owner - onCreate
2019-02-17 21:45:32.231 4664-4664/at.htl.lifecycleaware I/MainActivityObserver: Observer ON_CREATE
2019-02-17 21:45:32.232 4664-4664/at.htl.lifecycleaware I/MainActivity: Owner onStart
2019-02-17 21:45:32.232 4664-4664/at.htl.lifecycleaware I/MainActivityObserver: Observer ON_START
2019-02-17 21:45:32.236 4664-4664/at.htl.lifecycleaware I/MainActivity: Owner onResume
2019-02-17 21:45:32.236 4664-4664/at.htl.lifecycleaware I/MainActivityObserver: Observer ON_RESUME
2019-02-17 21:46:06.020 4664-4664/at.htl.lifecycleaware I/MainActivityObserver: Observer ON_PAUSE
2019-02-17 21:46:06.020 4664-4664/at.htl.lifecycleaware I/MainActivity: Owner onPause
2019-02-17 21:46:06.020 4664-4664/at.htl.lifecycleaware I/MainActivityObserver: Observer ON_STOP
2019-02-17 21:46:06.029 4664-4664/at.htl.lifecycleaware I/MainActivity: Owner onStop
2019-02-17 21:46:06.029 4664-4664/at.htl.lifecycleaware I/MainActivityObserver: Observer ON_DESTROY
2019-02-17 21:46:06.030 4664-4664/at.htl.lifecycleaware I/MainActivity: Owner onDestroy
2019-02-17 21:46:06.073 4664-4664/at.htl.lifecycleaware I/MainActivity: attached Observer to Owner - onCreate
2019-02-17 21:46:06.074 4664-4664/at.htl.lifecycleaware I/MainActivityObserver: Observer ON_CREATE
2019-02-17 21:46:06.076 4664-4664/at.htl.lifecycleaware I/MainActivity: Owner onStart
2019-02-17 21:46:06.076 4664-4664/at.htl.lifecycleaware I/MainActivityObserver: Observer ON_START
2019-02-17 21:46:06.079 4664-4664/at.htl.lifecycleaware I/MainActivity: Owner onResume
2019-02-17 21:46:06.079 4664-4664/at.htl.lifecycleaware I/MainActivityObserver: Observer ON_RESUME

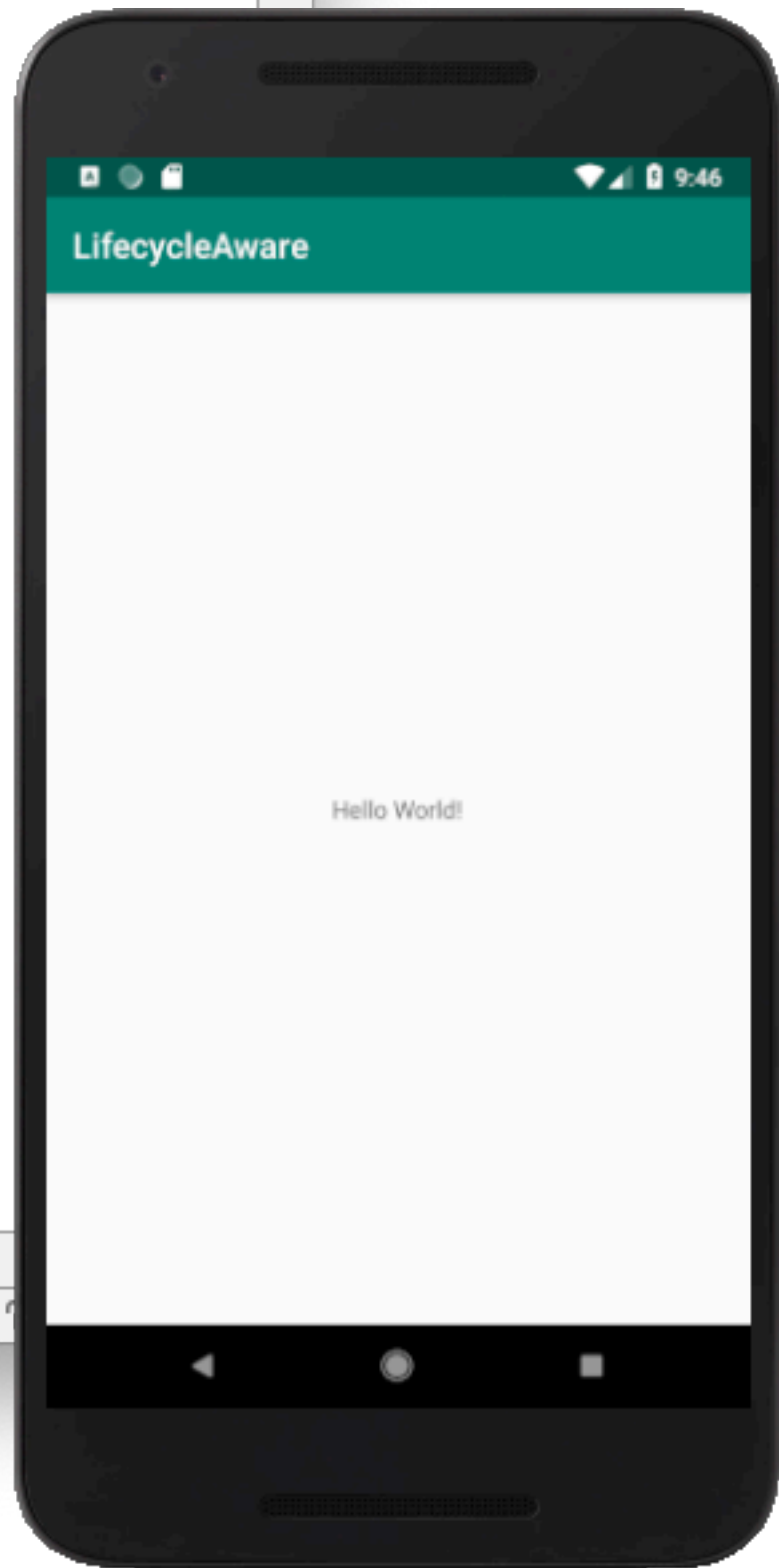
```

Install successful

Install successful (a minute ago)

39:1 LF UTF-8 4 spaces

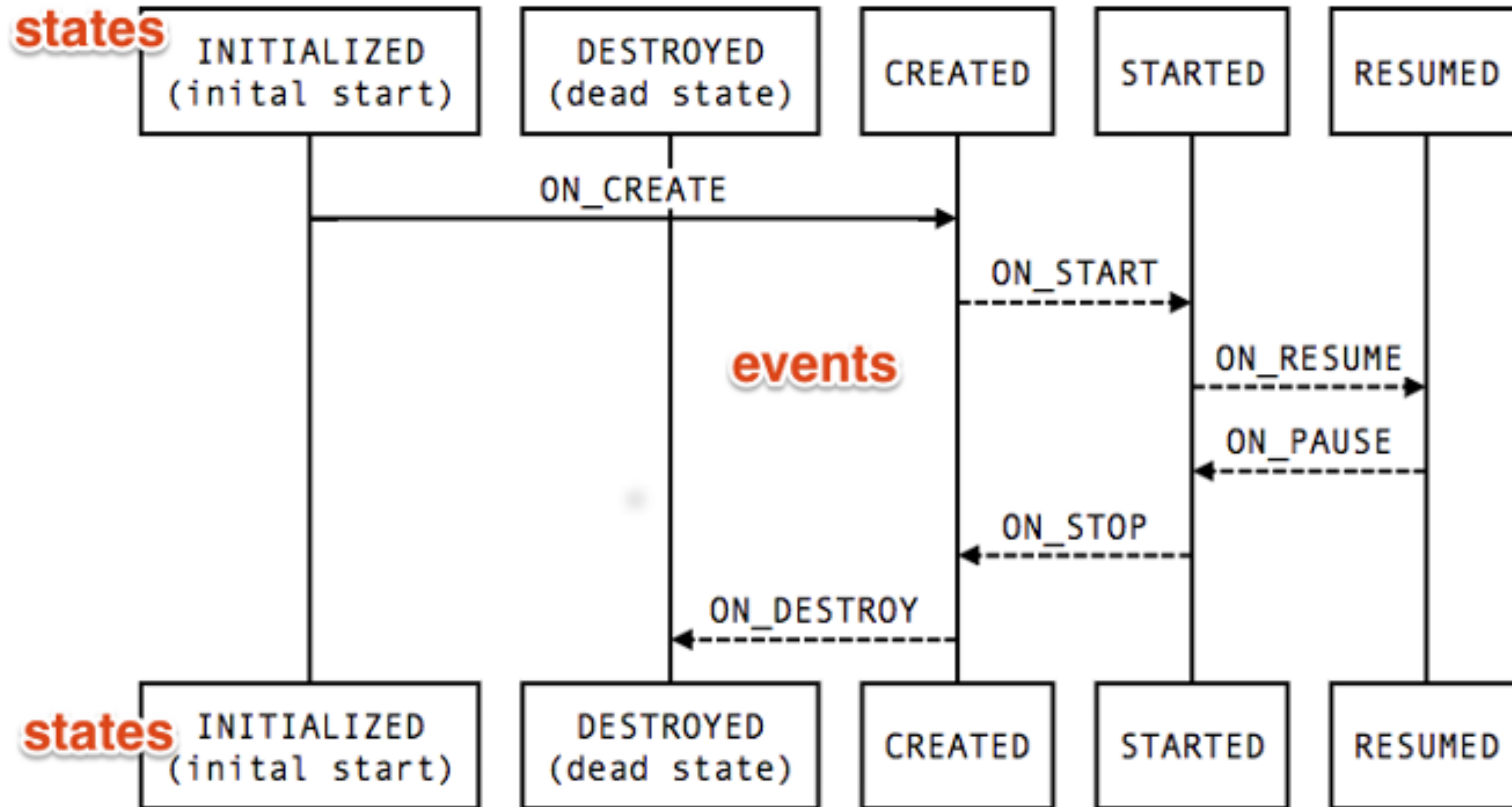
Funktioniert auch beim zweiten Mal



Zusammenfassung

- Lifecycle-Aware Components reagieren auf Änderungen des Zustands anderer UI-Komponenten
 - Eintragen der Dependency und des kapt-Compilers
 - Erstellen des Observer mit seinen @OnLifecycleEvent-Methoden
 - Registrieren des Observer beim Owner (der Activity oder dem Fragment)
 - Überschreiben der On-Methoden beim Owner

States and Events



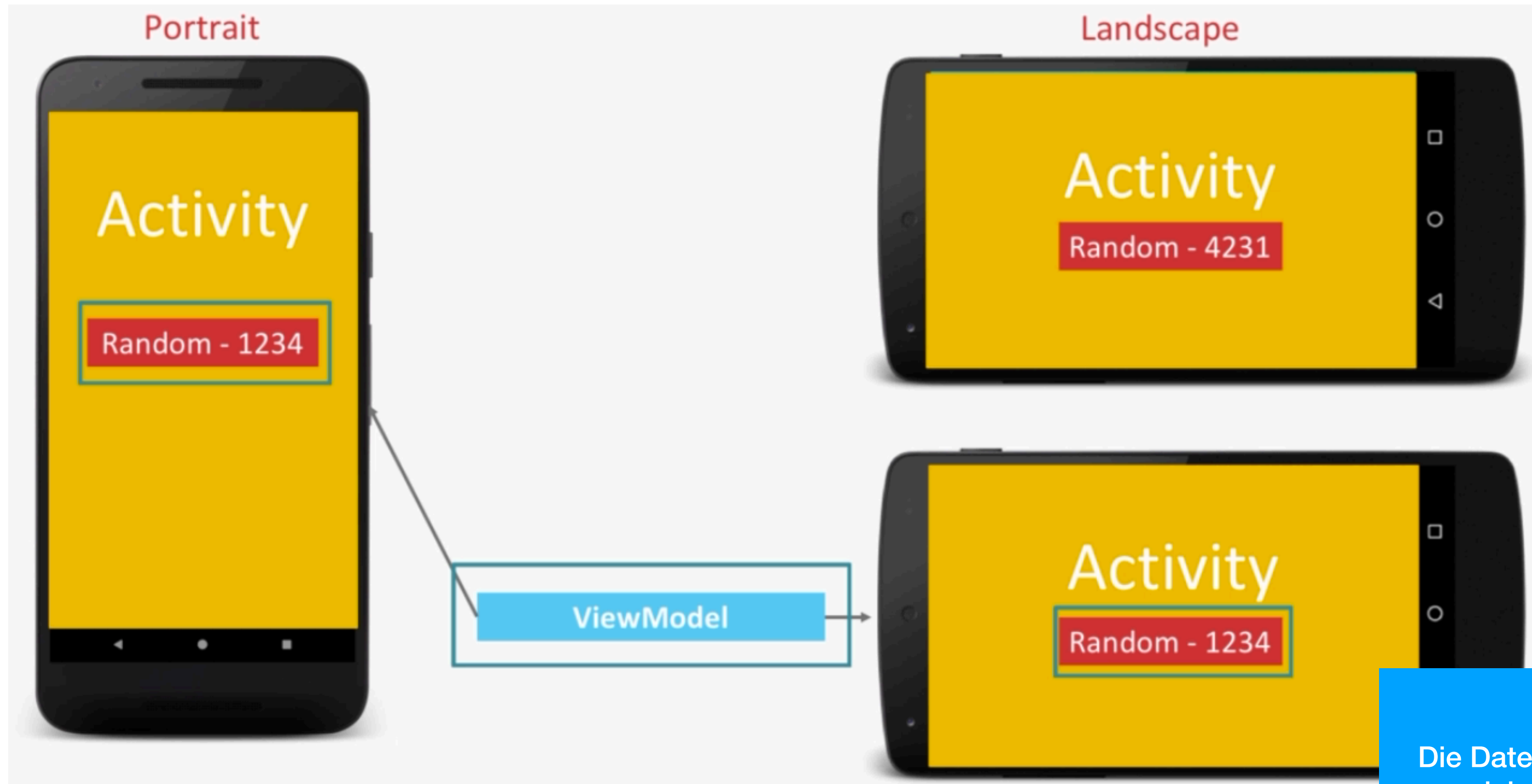
<https://developer.android.com/topic/libraries/architecture/lifecycle>

ViewModel

Problem

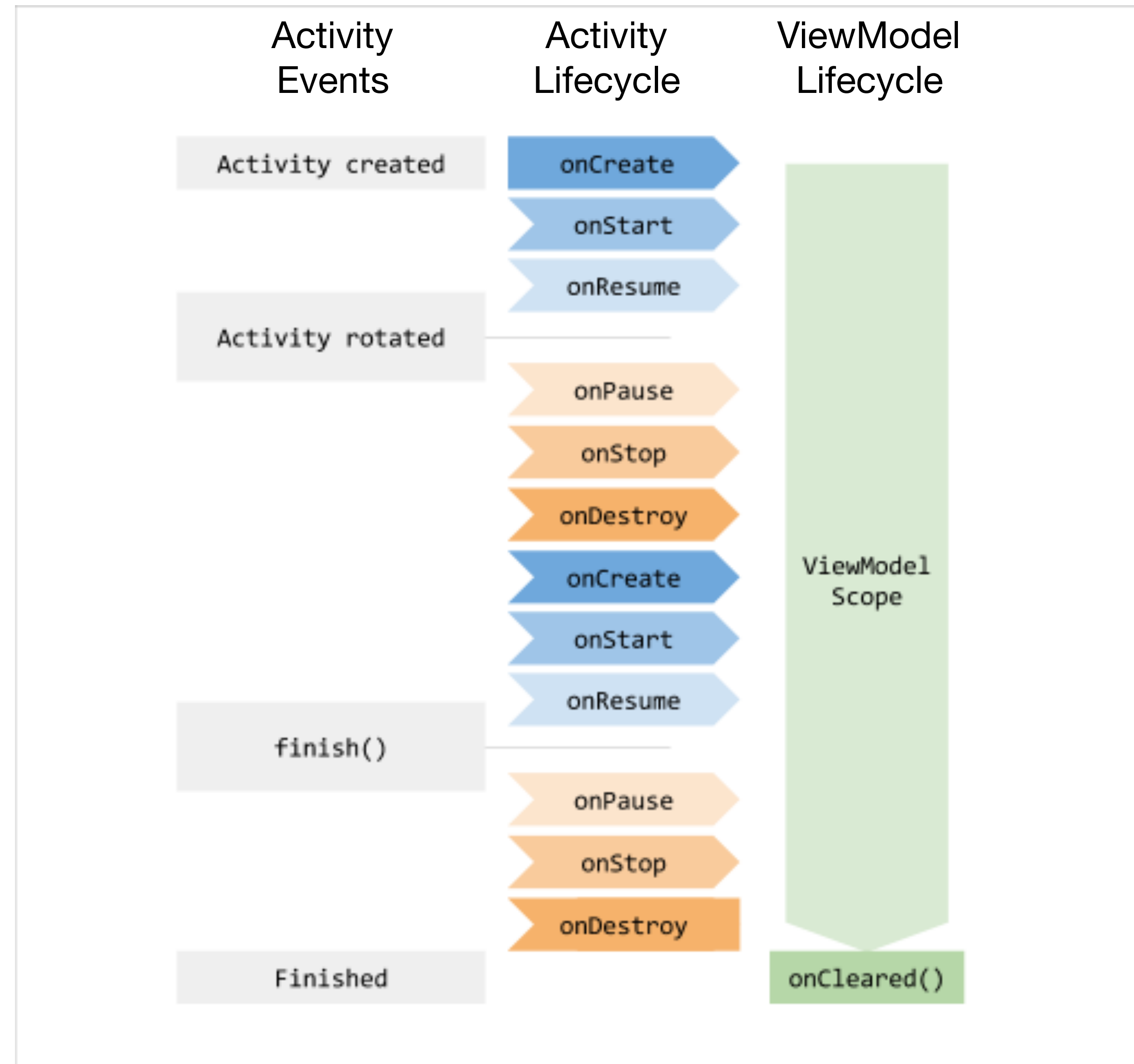


Lösung - ViewModel

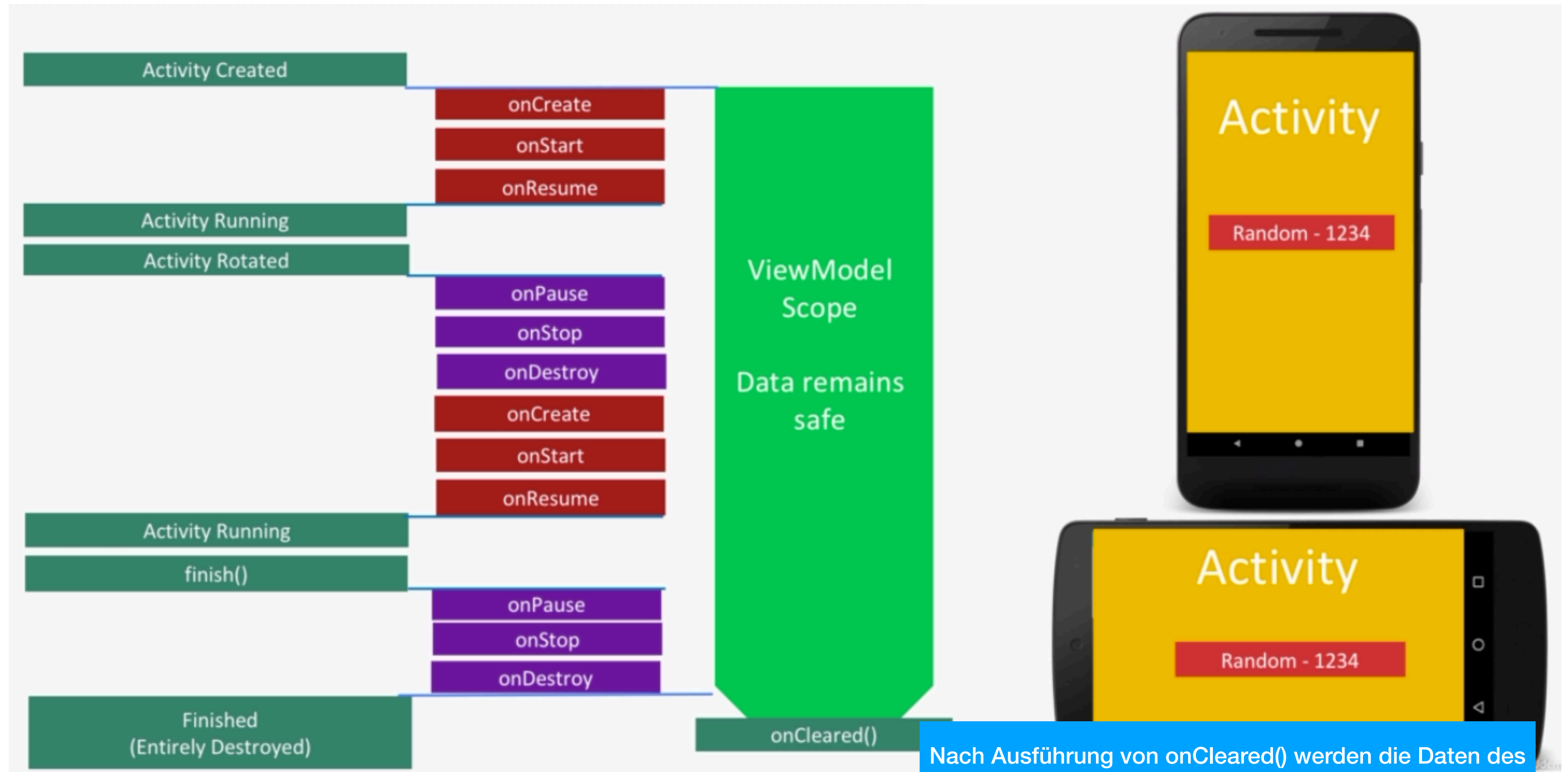


Die Daten werden im ViewModel gespeichert. Wird die neue Activity erstellt, so können diese gespeicherten Daten wiederverwendet werden.

Lebenszyklus

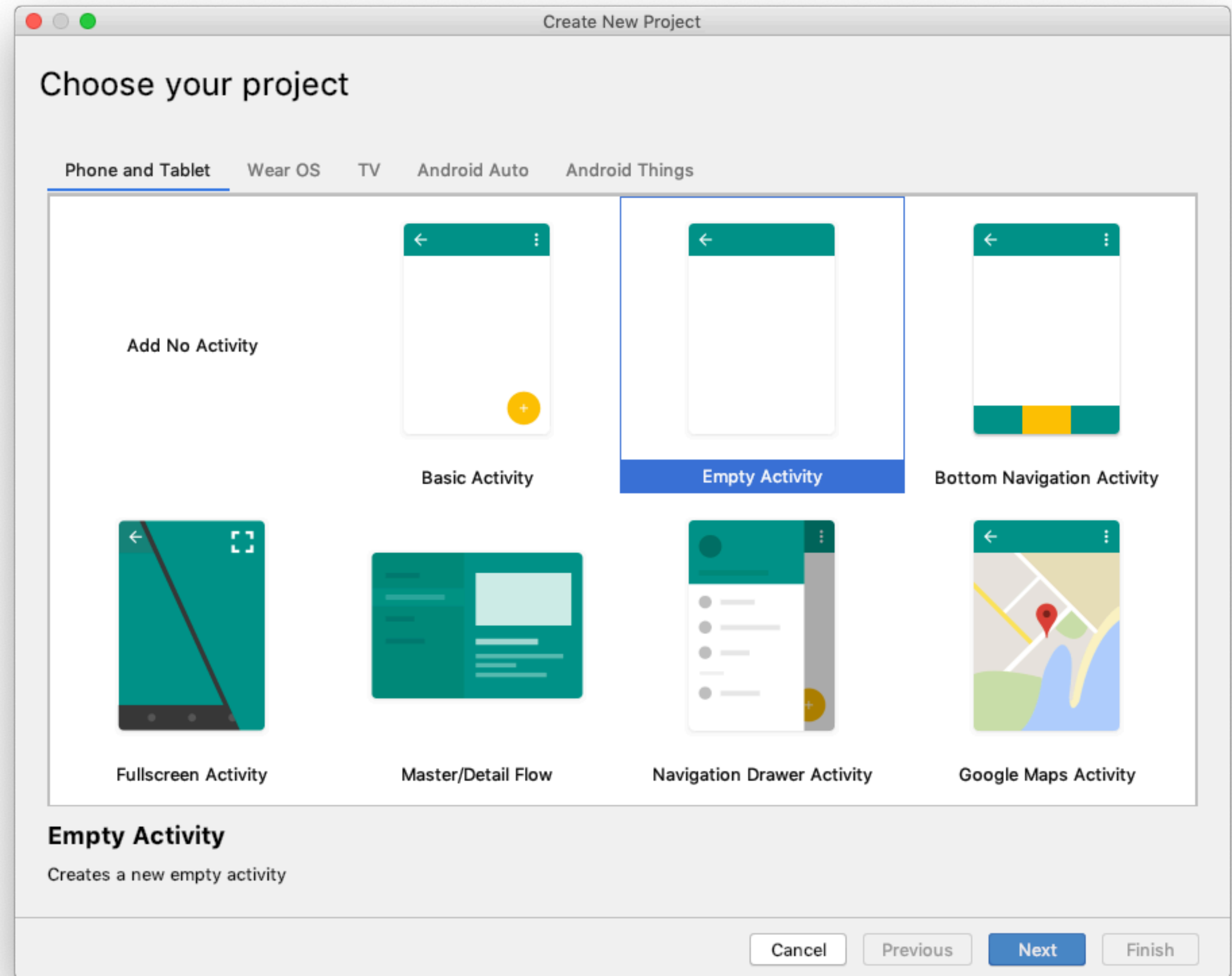


Beispiel



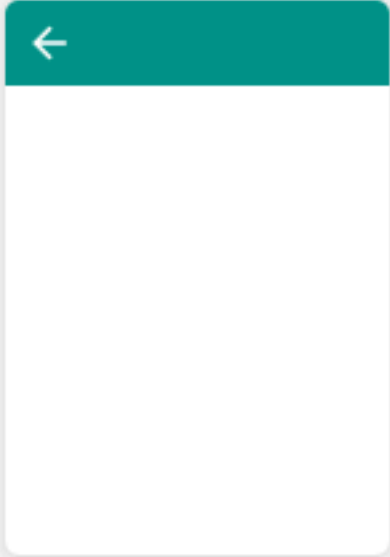
Nach Ausführung von `onCleared()` werden die Daten des ViewModels gelöscht

Variante 1: Ohne ViewModel



Create New Project

Configure your project



Empty Activity

Creates a new empty activity

Name
ViewModelLiveData

Package name
at.htl.viewmodellivedata

Save location
ndroid.kotlin/2019.jetpack.udemy/02.LifeCycle/ViewModelLiveData

Language
Kotlin

Minimum API level
API 23: Android 6.0 (Marshmallow)

i Your app will run on approximately **62.6%** of devices.
[Help me choose](#)

This project will support instant apps

Use AndroidX artifacts

Cancel Previous Next Finish

ViewModelLiveData [~/CloudStation/htl/skripten/themen/android.kotlin/2019.jetpack.udemy/02.LifeCycle/ViewModelLiveData] - .../app/src/main/java/at/htl/viewmodellivedata/MainActivity.kt [app]

ViewModelLiveData > app > src > main > java > at > htl > viewmodellivedata > MainActivity.kt

1 package at.htl.viewmodellivedata
2
3 import androidx.appcompat.app.AppCompatActivity
4 import android.os.Bundle
5
6 class MainActivity : AppCompatActivity() {
7
8 override fun onCreate(savedInstanceState: Bundle?) {
9 super.onCreate(savedInstanceState)
10 setContentView(R.layout.activity_main)
11 }
12 }
13

Build: Build Output x Sync x

- Build: completed successfully at 2019-02-18 09:03 909 ms
- Run build /Users/stuetz/CloudStation/htl/skripten/themen/android.kotlin/2019.jetpack.udemy/02.LifeCycle/ViewModelLiveData 587 ms
 - Load build 4 ms
 - Configure build 240 ms
 - Calculate task graph 87 ms
 - Run tasks 253 ms

Gradle build finished in 923 ms

13:1 LF UTF-8 4 spaces

The screenshot shows the Android Studio IDE with the following components:

- Project View:** Shows the project structure with folders for manifests, java, and res.
- Palette:** Lists various UI widgets like Button, ImageView, RecyclerView, etc.
- Design View:** Displays a visual representation of the layout with a TextView containing "Hello World!". A blue callout box with the text "Wir vergeben eine ID für die TextView" is overlaid on the design.
- Attributes Panel:** Shows the properties for the selected TextView. The `id` attribute is set to `tv_number` and is highlighted with a red box. Other attributes include `layout_width`, `layout_height`, and `text`.
- Component Tree:** Shows the hierarchy of the layout, including `ConstraintLayout` and the `Ab tv_number - "Hello World!"` TextView.

MainActivityDataGenerator.kt

```
1 package at.htl.viewmodellivedata
2
3 import android.util.Log
4 import kotlin.random.Random
5
6 class MainActivityDataGenerator {
7
8     private lateinit var myRandomNumber: String
9
10    fun getNumber(): String {
11        Log.i(TAG, "Get number")
12
13        if (!::myRandomNumber.isInitialized) {
14            this.createNumber()
15        }
16        return myRandomNumber
17    }
18
19    // kotlin.random.Random
20    private fun createNumber() {
21        Log.i(TAG, "Create new number")
22        myRandomNumber = "Number: " + (Random.nextInt(10 - 1) + 1)
23    }
24
25    companion object {
26        private val TAG = MainActivityDataGenerator::class.java.simpleName
27    }
28
29 }
```

(Note: In the original image, the line `myRandomNumber = "Number: " + (Random.nextInt(10 - 1) + 1)` in the `createNumber()` function is highlighted with a red box and a red arrow points to it.)

New Kotlin File/Class

Name: MainActivityDataGenerator

Kind: Class

Cancel OK

`::myRandomNumber` entspricht `this::myRandomNumber`. Der `::`-Operator dient zum Zugriff auf Klassenattribute (property reference)
<https://kotlinlang.org/docs/reference/reflection.html#property-references>
Der `::`-Operator konvertiert auch eine Methode in einen Lambda-Ausdruck ()
<https://stackoverflow.com/a/52463295>
<https://kotlinlang.org/docs/reference/reflection.html#function-references>

%

```
// java.util.Random
private fun createNumber() {
    Log.i(TAG, "Create new number")
    val random = java.util.Random()
    myRandomNumber = "Number: " + (random.nextInt(10 - 1) + 1)
}
```

:: - Operator

Function References

When we have a named function declared like this:

```
fun isOdd(x: Int) = x % 2 != 0
```

We can easily call it directly (`isOdd(5)`), but we can also use it as a function type value, e.g. pass it to another function. To do this, we use the `::` operator:

```
val numbers = listOf(1, 2, 3)
println(numbers.filter(::isOdd))
```

Target platform: JVM Running on kotlin v. 1.3.21

Here `::isOdd` is a value of function type `(Int) -> Boolean`.

<https://kotlinlang.org/docs/reference/reflection.html#function-references>

Property References

To access properties as first-class objects in Kotlin, we can also use the `::` operator:

```
val x = 1

fun main() {
    println(::x.get())
    println(::x.name)
}
```

Target platform: JVM Running on kotlin v. 1.3.21

The expression `::x` evaluates to a property object of type `KProperty<Int>`, which allows us to read its value using `get()` or retrieve the property name using the `name` property. For more information, please refer to the [docs on the KProperty class](#).

<https://kotlinlang.org/docs/reference/reflection.html#property-references>

MainActivity.kt

```
2
3  import androidx.appcompat.app.AppCompatActivity
4  import android.os.Bundle
5  import android.util.Log
6  import kotlinx.android.synthetic.main.activity_main.*
7
8  class MainActivity : AppCompatActivity() {
9
10     lateinit var data: MainActivityDataGenerator
11
12     override fun onCreate(savedInstanceState: Bundle?) {
13         super.onCreate(savedInstanceState)
14         setContentView(R.layout.activity_main)
15
16         data = MainActivityDataGenerator()
17         val myRandomNumber = data.getNumber()
18         tv_number.text = myRandomNumber
19
20         Log.i(TAG, "Random Number set")
21     }
22
23     companion object {
24         private val TAG = MainActivity::class.java.simpleName
25     }
26 }
27
```

Gradle build finished in 923 ms (53 minutes ago)

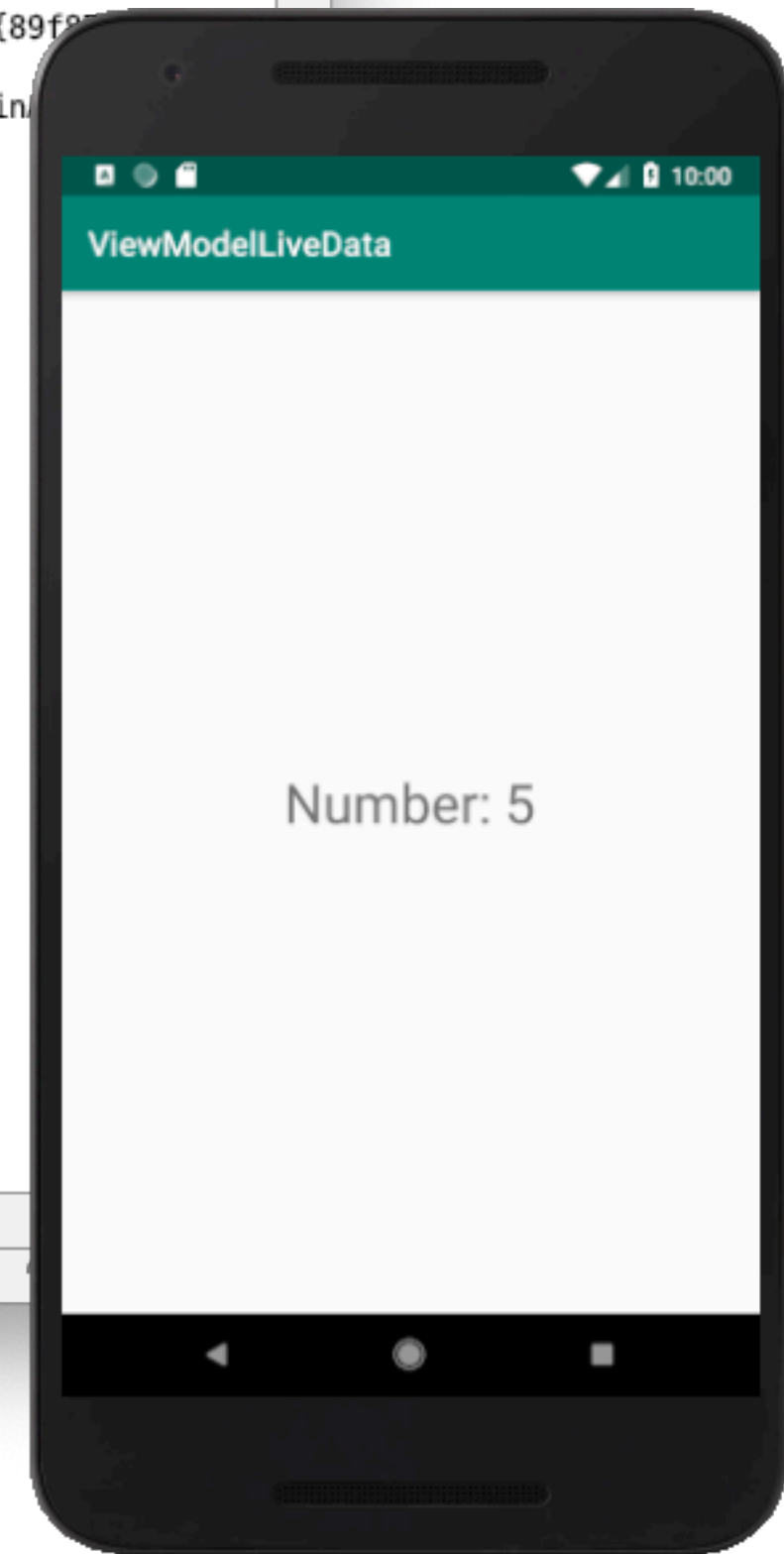
27:1 LF UTF-8 4 spaces

The screenshot shows an IDE window for a project named 'ViewModelLiveData'. The code editor displays the following Kotlin code for `MainActivityDataGenerator`:

```
class MainActivityDataGenerator {  
    private lateinit var myRandomNumber: String  
  
    fun getNumber(): String {
```

The Logcat window shows the following log messages:

```
2019-02-18 10:00:04.355 9851-9851/at.htl.viewmodellivedata I/zygote: at void at.htl.viewmodellivedata.MainActivity.onCreate(android.os.Bundle) (MainActivity.kt:14)  
2019-02-18 10:00:04.392 9851-9851/at.htl.viewmodellivedata I/MainActivityDataGenerator: Get number  
2019-02-18 10:00:04.392 9851-9851/at.htl.viewmodellivedata I/MainActivityDataGenerator: Create new number  
2019-02-18 10:00:04.393 9851-9851/at.htl.viewmodellivedata I/MainActivity: Random Number set  
2019-02-18 10:00:04.651 1682-1719/system_process I/ActivityManager: Displayed at.htl.viewmodellivedata/.MainActivity: +492ms
```



The screenshot shows an IDE with the following components:

- Code Editor:** Displays the Kotlin code for `MainActivityDataGenerator`. The class has a private property `myRandomNumber: String` and a `getNumber(): String` function.
- Logcat:** Shows a series of log messages. A red box highlights the following messages:

```
2019-02-18 10:00:59.678 9851-9851/at.htl.viewmodellivedata I/MainActivityDataGenerator: Get number
2019-02-18 10:00:59.678 9851-9851/at.htl.viewmodellivedata I/MainActivityDataGenerator: Create new number
2019-02-18 10:00:59.678 9851-9851/at.htl.viewmodellivedata I/MainActivity: Random Number set
```
- UI:** A blue text box at the bottom of the IDE contains the text: "Man sieht: Die „Daten“ gehen verloren und werden neu generiert".
- Emulator:** A tablet emulator on the right shows the app's UI with the text "Number: 2".

Variante 2: mit ViewModel

ViewModelLiveData [~/CloudStation/htl/skripten/themen/android.kotlin/2019.jetpack.udemy/02.LifeCycle/ViewModelLiveData] - app

Android

Project: ViewModelLiveData

app

Configure project in Project Structure dialog.

```
1  apply plugin: 'com.android.application'
2
3  apply plugin: 'kotlin-android'
4
5  apply plugin: 'kotlin-android-extensions'
6
7  apply plugin: 'kotlin-kapt'
8
9  android {
10     compileSdkVersion 28
11     defaultConfig {
12         applicationId "at.htl.viewmodellivedata"
13         minSdkVersion 23
14         targetSdkVersion 28
15         versionCode 1
16         versionName "1.0"
17         testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
18     }
19     buildTypes {
20         release {
21             minifyEnabled false
22             proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
23         }
24     }
25 }
26
27 dependencies {
28     implementation fileTree(dir: 'libs', include: ['*.jar'])
29     implementation "org.jetbrains.kotlin:kotlin-stdlib-jdk7:$kotlin_version"
30     implementation 'androidx.appcompat:appcompat:1.0.2'
31     implementation 'androidx.core:core-ktx:1.0.1'
32     implementation 'androidx.constraintlayout:constraintlayout:1.1.3'
33     testImplementation 'junit:junit:4.12'
34     androidTestImplementation 'androidx.test.ext:junit:1.1.0'
35     androidTestImplementation 'androidx.test.espresso:espresso-core:3.1.1'
36
37     def lifecycle_version = "2.0.0"
38     // ViewModel and LiveData
39     implementation "androidx.lifecycle:lifecycle-extensions:$lifecycle_version"
40
41 }
```

Gradle build finished in 575 ms (moments ago)

42:1 LF UTF-8 4 spaces

<https://developer.android.com/jetpack/androidx/releases/lifecycle>

Erstellen eines ViewModels

```
class MainActivityDataGenerator: ViewModel() {  
  
    private lateinit var myRandomNumber: String  
  
    fun getNumber(): String {  
        Log.i(TAG, "Get number")  
  
        if (!::myRandomNumber.isInitialized) {  
            this.createNumber()  
        }  
        return myRandomNumber  
    }  
  
    // kotlin.random.Random  
    private fun createNumber() {  
        Log.i(TAG, "Create new number")  
        myRandomNumber = "Number: " + (Random.nextInt(10 - 1) + 1)  
    }  
  
    companion object {  
        private val TAG = MainActivityDataGenerator::class.java.simpleName  
    }  
}
```

Wir verwenden den bereits bestehenden MainActivityDataGenerator als ViewModel

ViewModel in MainActivity verwenden

```
class MainActivity : AppCompatActivity() {  
  
    // lateinit var data: MainActivityDataGenerator  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_main)  
  
        val model = ViewModelProviders  
            .of(this)  
            .get(MainActivityDataGenerator::class.java)  
        val myRandomNumber = model.getNumber()  
  
        //data = MainActivityDataGenerator()  
        //val myRandomNumber = data.getNumber()  
        tv_number.text = myRandomNumber  
  
        Log.i(TAG, "Random Number set")  
    }  
  
    companion object {  
        private val TAG = MainActivity::class.java.simpleName  
    }  
}
```

Beachte: Verwende [ViewModelProviders](#)

ViewModelProviders (androidx.life
ViewModelProvider (androidx.li...
Did you know that Quick Documentation View (F1) works in completion lookups as well? >>

The screenshot shows the Android Studio IDE with the following components:

- Code Editor:** Displays Kotlin code for MainActivity.kt, showing lines 18 and 19: `.of(this)` and `.get(MainActivityDataGenerator::class.java)`.
- Logcat:** Shows a log entry for MainActivityDataGenerator: `I/MainActivityDataGenerator: Create new number`, which is highlighted with a red box.
- Callout:** A blue box with the text "Eine neue Zufallszahl wird generiert" (A new random number is generated).
- Emulator:** Displays the app's UI with the text "Number: 6".
- Bottom Bar:** Shows "Install successful" and "Gradle build finished in 1 m 47 s 313 ms".

The screenshot shows the Android Studio IDE with the following components:

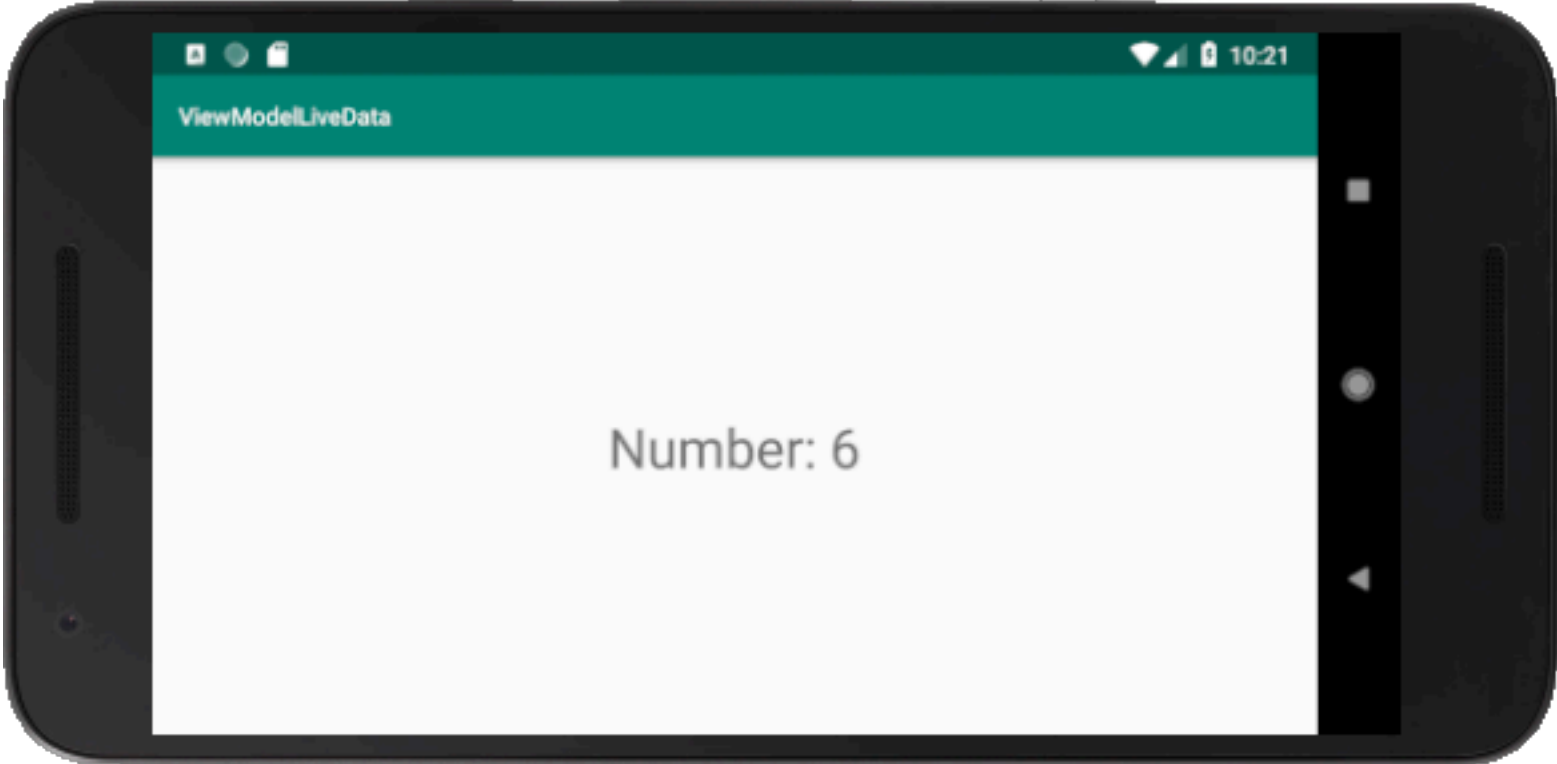
- Project Explorer:** Shows the project structure with folders for `manifests`, `java`, and `at.htl.viewmodellivedata`.
- Code Editor:** Displays the `MainActivity.kt` file with lines 18 and 19 visible. Line 19 contains the code `.get(MainActivityDataGenerator::class.java)`.
- Logcat:** Shows a list of log messages for the application. The following messages are highlighted with a red box:

```
2019-02-18 10:20:29.843 10616-10616/at.htl.viewmodellivedata I/MainActivity: Random Number set
2019-02-18 10:21:04.279 10616-10616/at.htl.viewmodellivedata I/MainActivityDataGenerator: Get number
2019-02-18 10:21:04.279 10616-10616/at.htl.viewmodellivedata I/MainActivity: Random Number set
```
- Device View:** Shows a virtual Android phone displaying the application's UI with the text "Number: 6".

Logcat Message Details:

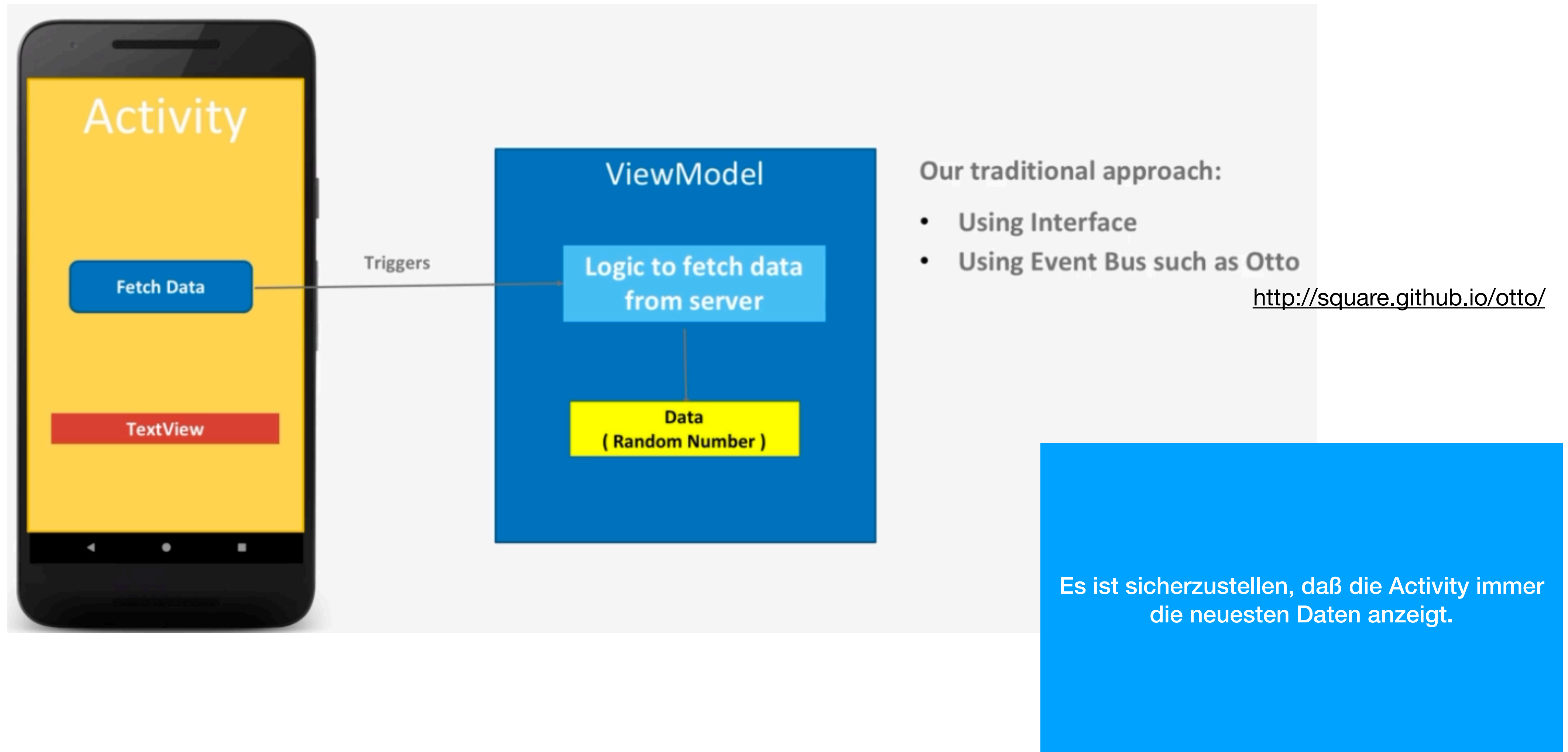
| Time | Process | Thread | Message |
|-------------------------|--------------------------------------|------------------------------|--|
| 2019-02-18 10:20:29.789 | 10616-10616/at.htl.viewmodellivedata | I/zygote: | at void at.htl.viewmodellivedata.MainActivity.onCreate(android.os.Bundle) (MainActivity.kt:15) |
| 2019-02-18 10:20:29.789 | 10616-10616/at.htl.viewmodellivedata | I/zygote: | at void at.htl.viewmodellivedata.MainActivity.onCreate(android.os.Bundle) (MainActivity.kt:15) |
| 2019-02-18 10:20:29.790 | 10616-10616/at.htl.viewmodellivedata | I/zygote: | at void at.htl.viewmodellivedata.MainActivity.onCreate(android.os.Bundle) (MainActivity.kt:15) |
| 2019-02-18 10:20:29.790 | 10616-10616/at.htl.viewmodellivedata | I/zygote: | at void at.htl.viewmodellivedata.MainActivity.onCreate(android.os.Bundle) (MainActivity.kt:15) |
| 2019-02-18 10:20:29.790 | 10616-10616/at.htl.viewmodellivedata | I/zygote: | at void at.htl.viewmodellivedata.MainActivity.onCreate(android.os.Bundle) (MainActivity.kt:15) |
| 2019-02-18 10:20:29.791 | 10616-10616/at.htl.viewmodellivedata | I/zygote: | at void at.htl.viewmodellivedata.MainActivity.onCreate(android.os.Bundle) (MainActivity.kt:15) |
| 2019-02-18 10:20:29.842 | 10616-10616/at.htl.viewmodellivedata | I/MainActivityDataGenerator: | Get number |
| 2019-02-18 10:20:29.842 | 10616-10616/at.htl.viewmodellivedata | I/MainActivityDataGenerator: | Create new number |
| 2019-02-18 10:20:29.843 | 10616-10616/at.htl.viewmodellivedata | I/MainActivity: | Random Number set |
| 2019-02-18 10:21:04.279 | 10616-10616/at.htl.viewmodellivedata | I/MainActivityDataGenerator: | Get number |
| 2019-02-18 10:21:04.279 | 10616-10616/at.htl.viewmodellivedata | I/MainActivity: | Random Number set |

Die bestehende Zufallszahl wird wiederverwendet.
Die Daten gehen nicht verloren.

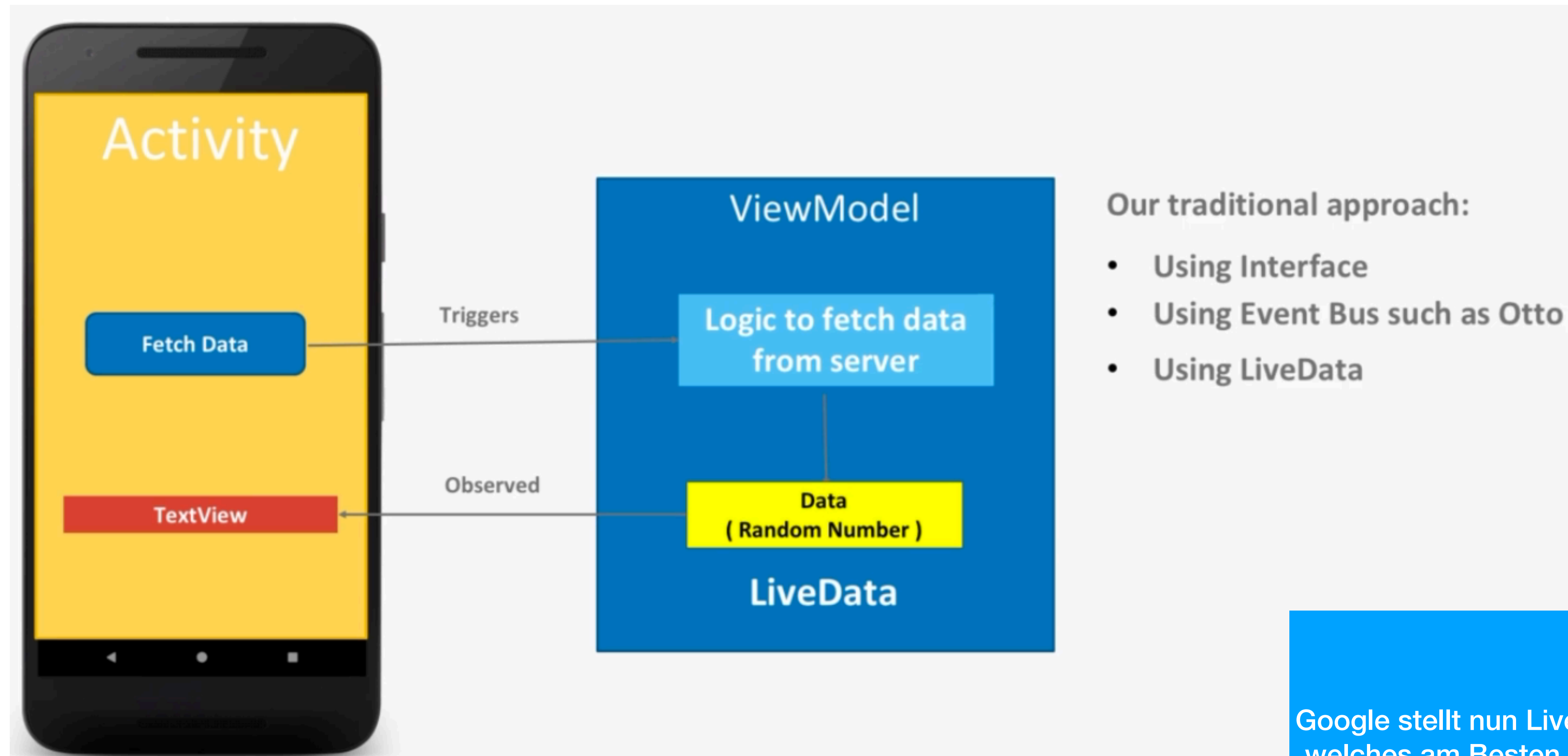


LiveData

Problem



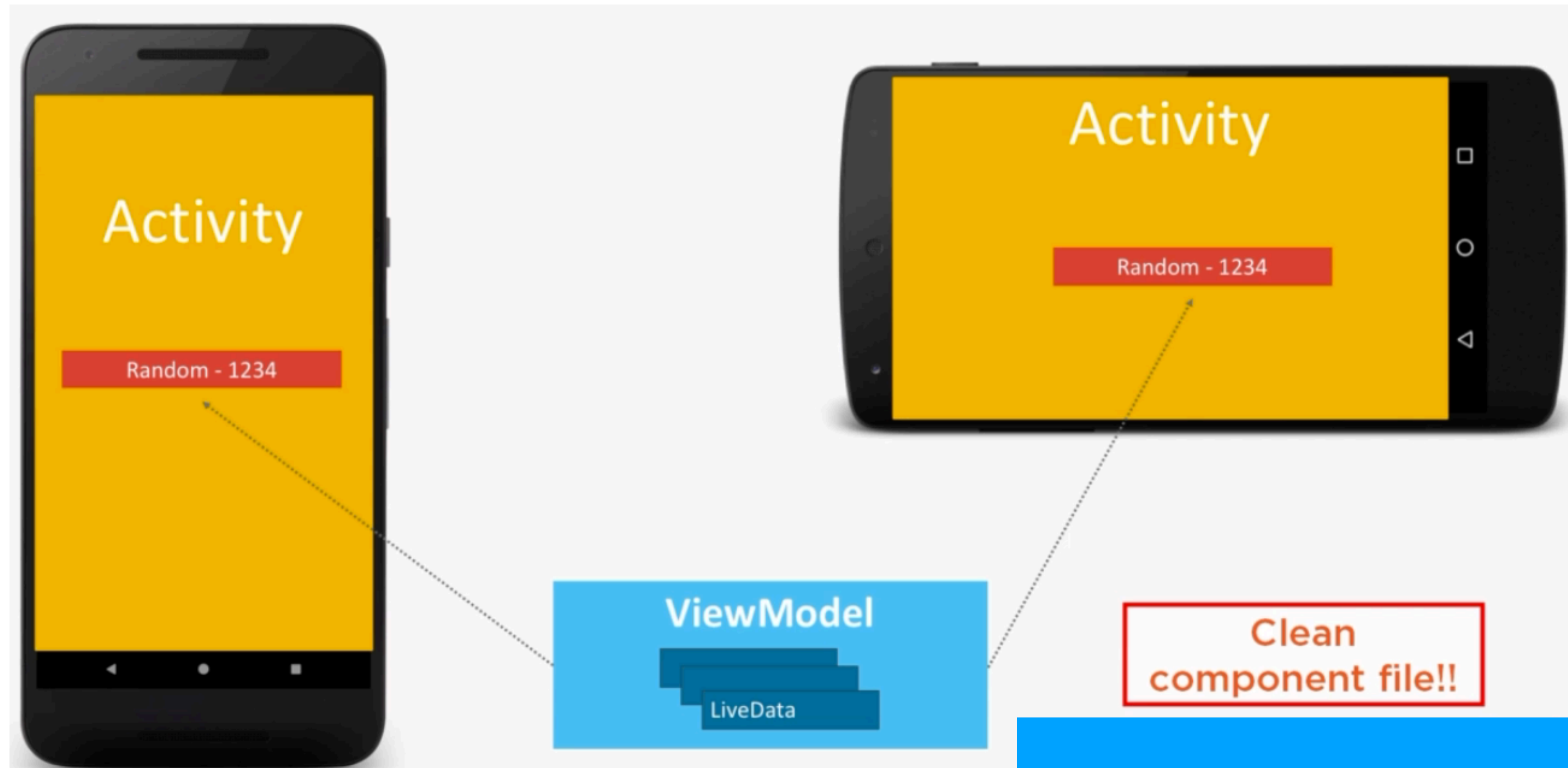
Lösung



Our traditional approach:

- Using Interface
- Using Event Bus such as Otto
- Using LiveData

Google stellt nun LiveData zur Verfügung, welches am Besten mit dem ViewModel kombiniert wird.

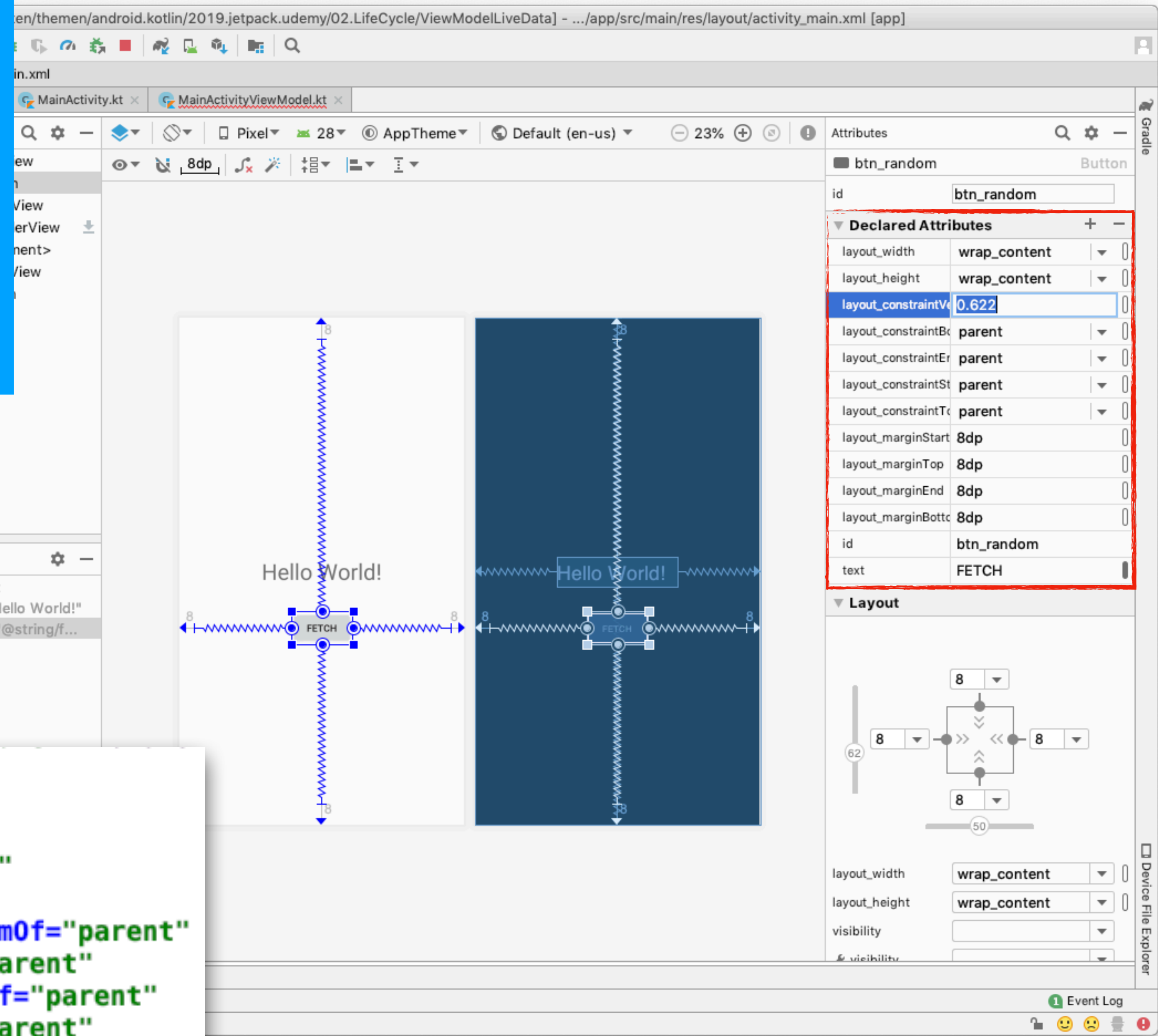


Die Activities und Fragments enthalten keinen Code zur Aktualisierung der Daten. Sie sind nur für die Anzeige der Daten verantwortlich

Zunächst führen wir ein paar Änderungen an unserem Beispiel durch:

Zunächst erstellen wir einen neuen Button.

Dabei wird eine String-Resource erstellt.



```
<Button
    android:id="@+id/btn_random"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/fetch"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.622"/>
```

MainActivityViewModel.kt

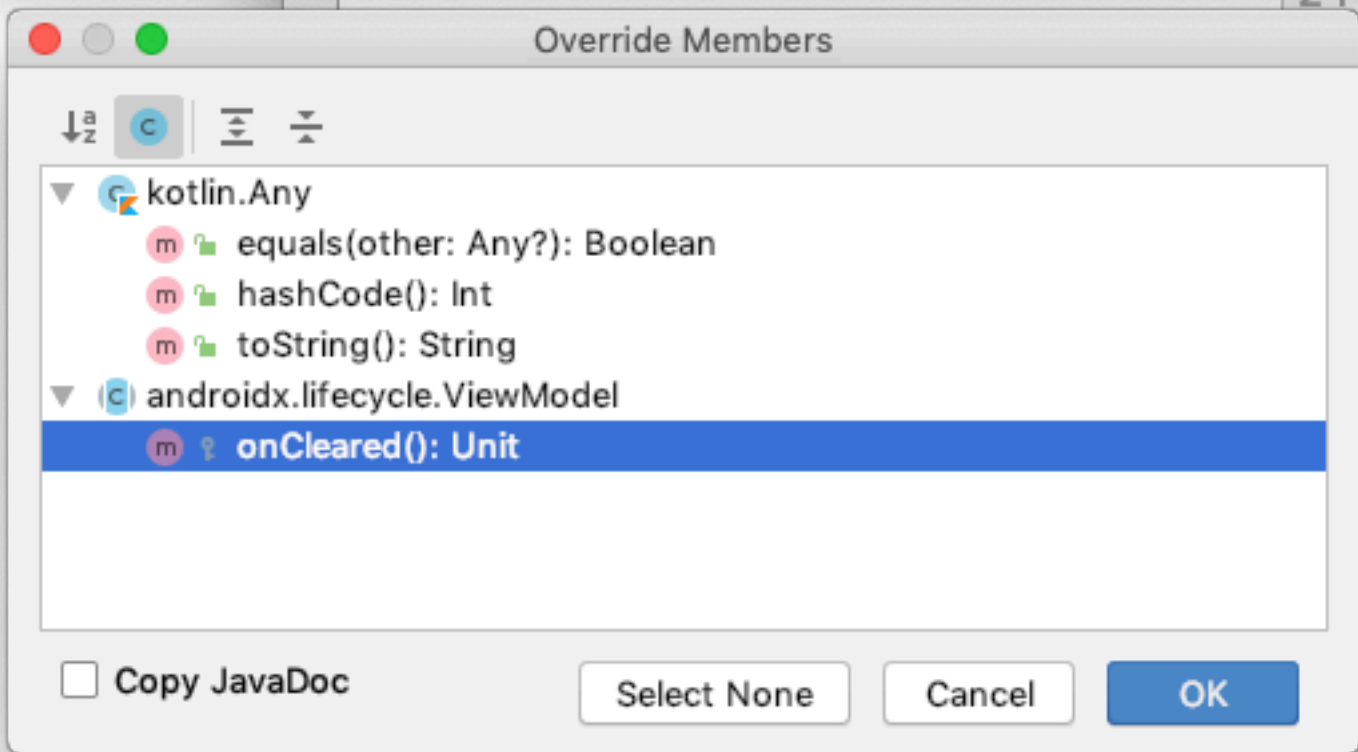
Wir verwenden MutableLiveData, da wir keine Zugriff auf Methoden freigeben, die die Daten ändern könnten. MutableLiveData wird von LiveData abgeleitet

```
1 package at.htl.viewmodellivedata
2
3 import ...
4
5 class MainActivityViewModel: ViewModel() {
6
7     private lateinit var myRandomNumber: MutableLiveData<String>
8     //private lateinit var myRandomNumber: String
9
10    fun getNumber(): MutableLiveData<String> {
11        Log.i(TAG, "Get number")
12
13        if (!::myRandomNumber.isInitialized) {
14            myRandomNumber = MutableLiveData()
15            this.createNumber()
16        }
17        return myRandomNumber
18    }
19
20    fun createNumber() {
21        Log.i(TAG, "Create new number")
22        myRandomNumber.value = "Number: " + (Random.nextInt(10 - 1) + 1)
23
24        // myRandomNumber = "Number: " + (Random.nextInt(10 - 1) + 1)
25    }
26
27    override fun onCleared() {
28        super.onCleared()
29        Log.i(TAG, "ViewModel destroyed")
30    }
31
32    companion object {
33        private val TAG = MainActivityViewModel::class.java.simpleName
34    }
35}
```

```
package androidx.lifecycle;
/**
 * {@link LiveData} which publicly exposes {@link #setValue(T)}
 *
 * @param <T> The type of data hold by this instance
 */
@WeakerAccess
public class MutableLiveData<T> extends LiveData<T> {
    @Override
    public void postValue(T value) { super.postValue(value); }

    @Override
    public void setValue(T value) { super.setValue(value); }
}
```

Override Methods via ^O (Ctrl+O for Win/Linux)



MainActivity.kt

```
package at.htl.viewmodellivedata

import ...

class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        val model = ViewModelProviders
            .of(this)
            .get(MainActivityViewModel::class.java)
        val myRandomNumber = model.getNumber()

        myRandomNumber.observe(this, Observer<String> { number ->
            tv_number.text = number
            Log.i(TAG, "Random number set")
        })

        btn_random.setOnClickListener {
            model.createNumber()
        }
    }

    companion object {
        private val TAG = MainActivity::class.java.simpleName
    }
}
```

die MainActivity beobachtet das
ViewModel und aktualisiert die
View

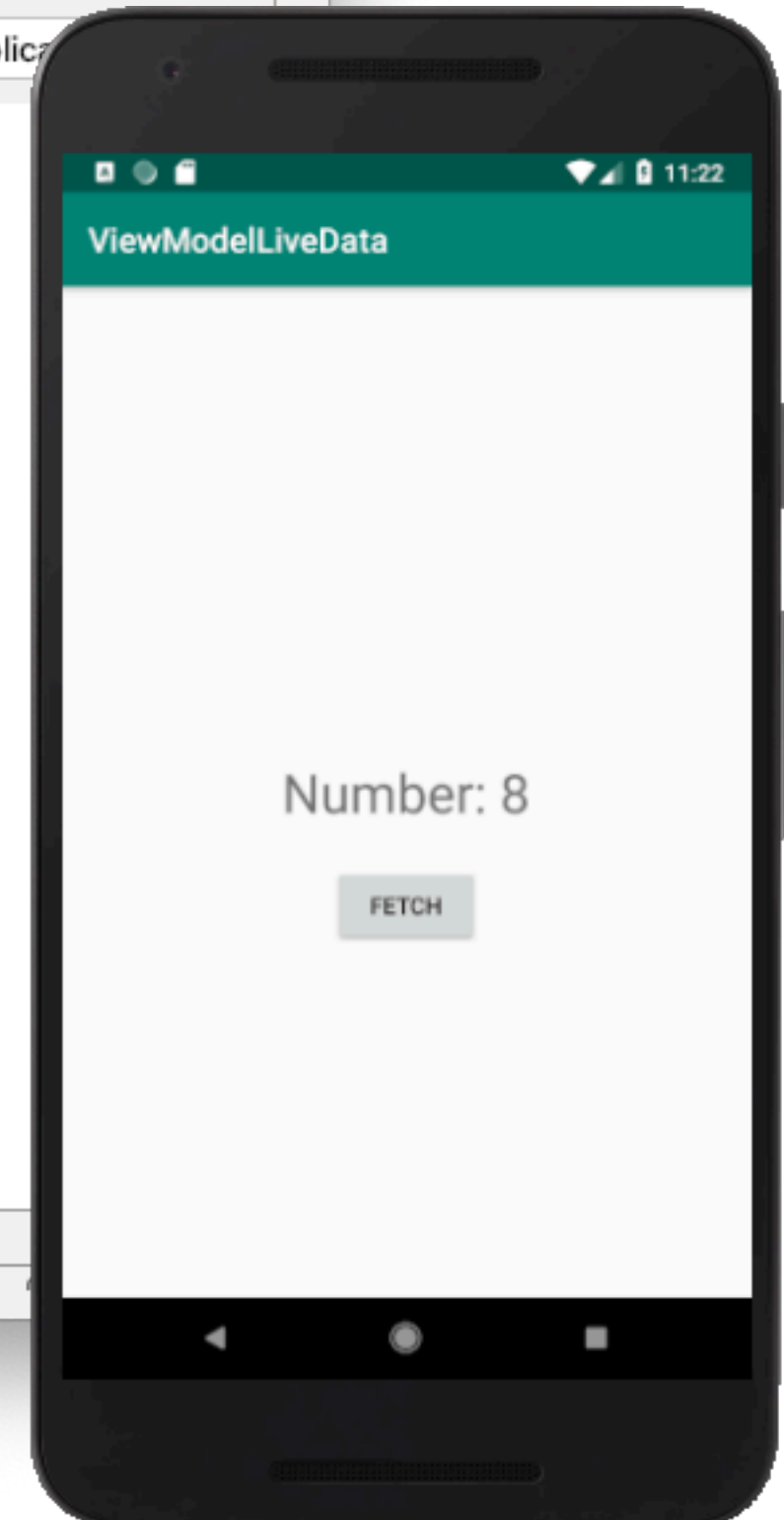
onClickListener für den Button


```
1 package at.htl.viewmodellivedata
2
3 import ...
7
8 class MainActivityViewModel: ViewModel() {
9
10     private lateinit var myRandomNumber: MutableLiveData<String>
11     //private lateinit var myRandomNumber: String
12
13     fun getNumber(): MutableLiveData<String> {
14         Log.i(TAG, "Get number")
15
16         if (!myRandomNumber.isInitialized) {
```

Logcat output:

```
2019-02-18 11:21:20.510 11532-11532/at.htl.viewmodellivedata I/zygote: at void at.htl.viewmodellivedata.MainActivity.onCreate(android.os.Bundle) (MainActivity.kt:14)
2019-02-18 11:21:20.510 11532-11532/at.htl.viewmodellivedata I/zygote: at void at.htl.viewmodellivedata.MainActivity.onCreate(android.os.Bundle) (MainActivity.kt:14)
2019-02-18 11:21:20.511 11532-11532/at.htl.viewmodellivedata I/zygote: at void at.htl.viewmodellivedata.MainActivity.onCreate(android.os.Bundle) (MainActivity.kt:14)
2019-02-18 11:21:20.511 11532-11532/at.htl.viewmodellivedata I/zygote: at void at.htl.viewmodellivedata.MainActivity.onCreate(android.os.Bundle) (MainActivity.kt:14)
2019-02-18 11:21:20.512 11532-11532/at.htl.viewmodellivedata I/zygote: at void at.htl.viewmodellivedata.MainActivity.onCreate(android.os.Bundle) (MainActivity.kt:14)
2019-02-18 11:21:20.513 11532-11532/at.htl.viewmodellivedata I/zygote: at void at.htl.viewmodellivedata.MainActivity.onCreate(android.os.Bundle) (MainActivity.kt:14)
2019-02-18 11:21:20.611 11532-11532/at.htl.viewmodellivedata I/MainActivityViewModel: Get number
2019-02-18 11:21:20.611 11532-11532/at.htl.viewmodellivedata I/MainActivityViewModel: Create new number
2019-02-18 11:21:20.619 11532-11532/at.htl.viewmodellivedata I/MainActivity: Random number set
```

Nach dem Starten wird die Zahl neu generiert



The screenshot shows an IDE window for a project named 'ViewModelLiveData'. The main editor displays the Kotlin code for 'MainActivityViewModel.kt':

```
1 package at.htl.viewmodellivedata
2
3 import ...
7
8 class MainActivityViewModel: ViewModel() {
9
10     private lateinit var myRandomNumber: MutableLiveData<String>
11     //private lateinit var myRandomNumber: String
12
13     fun getNumber(): MutableLiveData<String> {
14         Log.i(TAG, "Get number")
15
16         if (!myRandomNumber.isInitialized) {
```

The Logcat window below shows the following log entries:

```
2019-02-18 11:21:20.510 11532-11532/at.htl.viewmodellivedata I/zygote: at void at.htl.viewmodellivedata.MainActivity.onCreate(android.os.Bundle) (MainActivity.kt:14)
2019-02-18 11:21:20.510 11532-11532/at.htl.viewmodellivedata I/zygote: at void at.htl.viewmodellivedata.MainActivity.onCreate(android.os.Bundle) (MainActivity.kt:14)
2019-02-18 11:21:20.511 11532-11532/at.htl.viewmodellivedata I/zygote: at void at.htl.viewmodellivedata.MainActivity.onCreate(android.os.Bundle) (MainActivity.kt:14)
2019-02-18 11:21:20.511 11532-11532/at.htl.viewmodellivedata I/zygote: at void at.htl.viewmodellivedata.MainActivity.onCreate(android.os.Bundle) (MainActivity.kt:14)
2019-02-18 11:21:20.512 11532-11532/at.htl.viewmodellivedata I/zygote: at void at.htl.viewmodellivedata.MainActivity.onCreate(android.os.Bundle) (MainActivity.kt:14)
2019-02-18 11:21:20.513 11532-11532/at.htl.viewmodellivedata I/zygote: at void at.htl.viewmodellivedata.MainActivity.onCreate(android.os.Bundle) (MainActivity.kt:14)
2019-02-18 11:21:20.611 11532-11532/at.htl.viewmodellivedata I/MainActivityViewModel: Get number
2019-02-18 11:21:20.611 11532-11532/at.htl.viewmodellivedata I/MainActivityViewModel: Create new number
2019-02-18 11:21:20.619 11532-11532/at.htl.viewmodellivedata I/MainActivity: Random number set
2019-02-18 11:23:12.684 11532-11532/at.htl.viewmodellivedata I/MainActivityViewModel: Create new number
2019-02-18 11:23:12.684 11532-11532/at.htl.viewmodellivedata I/MainActivity: Random number set
```

A blue callout box contains the text: "Durch Klicken des Button wird eine neue Zahl generiert und automatisch in der Activity dargestellt."

On the right, a smartphone emulator displays the app's UI with the title 'ViewModelLiveData' and the text 'Number: 4' above a 'FETCH' button.

Wie gehts weiter?

- <https://codelabs.developers.google.com/codelabs/android-room-with-a-view/#0>
-



Noch
Fragen?

Source

<https://www.udemy.com/android-jetpack-architecture-components/>

The screenshot shows the UdeMy website interface. At the top, there's a navigation bar with the UdeMy logo, a search bar, and links for 'UdeMy for Business', 'Become an instructor', 'Log In', and 'Sign Up'. Below this is a breadcrumb trail: 'Development > Mobile Apps > Android Game Development'. The main content area features the course title 'Android Jetpack Architecture Components' in large white text on a dark background. Below the title is a subtitle: 'Utilize Android Jetpack Architecture components to make your Android application development flexible and maintainable'. There are also indicators for 'NEW', a star rating of 0.0 (0 ratings), and '4 students enrolled'. A 'Preview this course' button is visible on the right side of the course card.



What you'll learn

- ✓ Get introduced to Android architecture components
- ✓ Provide stability in your app by handling life cycles, view models, and live data
- ✓ Load data gradually and gracefully in RecyclerView by using the Paging library
- ✓ Explore how to perform CRUD operations in the Room database
- ✓ Use the Data Binding library to bind data to the UI
- ✓ Implement effective in-app navigation by using the Navigation architecture component
- ✓ Implement a local database to store structured data by using the Room database
- ✓ Schedule tasks asynchronously by using Work Manager

€10.99 ~~€124.99~~ 91% off
🕒 5 hours left at this price!

[Add to cart](#)

[Buy now](#)

30-Day Money-Back Guarantee

Includes

- 📺 3 hours on-demand video
- 📄 1 downloadable resource
- 🌐 Full lifetime access
- 📱 Access on mobile and TV
- 🏆 Certificate of Completion