

[illegible]

```

1 import scala.collection.mutable.HashMap
2
3 // A custom abstract generic class resembling @State (see also @Optionalify). //
4 import scala.concurrent.ExecutionContext
5
6 public abstract class MyObject { // Default to Object's constructor.
7   private static final MyObject NONE_obj = new MyObject() // singleton new instance.
8
9   public static cb MyObject none() { // factory method
10     @SuppressWarnings("unchecked")
11     // Non-nullable HashMap KHM field contains no element so narrowing it as type MyObject
12     = from MyObject<Object> won't cause heap pollution and is thus safe.
13     // narrowing returns type consistency in MyObject of factory method. This aligns cb MyObject.
14
15     MyObject temp = (MyObject) NONE_obj;
16     return temp;
17   }
18
19   public static cb MyObject none(content) // factory method
20     // returns type can't be NoneObj
21     // given private static (abstractly top-level) nested class SomeObj.
22
23     return new SomeObj(content);
24   }
25
26   public static cb MyObject of(content) {
27     return (content == null) ? MyObject.none() : new SomeObj(content);
28   }
29 }
30
31 protected abstract T get();
32
33 public abstract MyObject filter(predicateCondition:() => predicate): // PRED
34
35 // MyObj-map consumes (inputs) list (args and produces (outputs) 2nd param.
36 public abstract cb MyObject map(transformer:() => T, to extends to mapObj): // PRED
37
38 public abstract T orElse(): defaultContent();
39
40 public abstract T orElseIf(predicate:() => PRED)
41
42 public abstract void ifPresent(Consumer<()=> to consumer): // PRED
43
44 public abstract void ifPresentOrElse(Consumer<()=> to consumer, ()=> PRED)
45
46 private static final class SomeObj extends MyObject{
47   // Non-generic concrete static nested class. Default to Object's constructor.
48   @Override
49   protected Object obj() throws RuntimeException {
50     throw new RuntimeException("not implemented");
51   }
52 }
53
54 @Override
55 public MyObject filter(predicateCondition:()=> super Object predicate) {
56   return this; // unconditionally true
57 }
58
59 @Override
60 public cb MyObject map(transform:()=> super Object, to extends to mapObj) {
61   return MyObject.none(); // unconditionally true, cannot be this type-erased to generic MyObject
62 }
63
64 @Override
65 public cb MyObject flatMap(transform:()=> super Object,
66   to extends MyObject extends cb flatMapObj) {
67   return MyObject.none(); // unconditionally true, cannot be this type-erased to generic MyObject
68 }
69
70 @Override
71 public Object orElse(defaultContent) {
72   return defaultContent();
73 }
74
75 @Override
76 public Object orElseIf(predicate:()=> extends Object producer) // PRED
77   return producer.produce();
78 }
79
80 @Override
81 public void ifPresent(Consumer<()=> super Object consumer) // PRED
82   return;
83 }
84
85 public boolean equals(Object other) {
86   // Unlike new instance exists referenced by NONE_obj always.
87   return other instanceof Some;
88 }
89
90 @Override
91 public String toString() {
92   return "T";
93 }
94
95 private static final class SomeObj extends MyObject {
96   // Generic concrete static nested class.
97   // Public methods in a private nested class accessible by only enclosing MyObject's class.
98   private final Content;
99
100   //
101   = Constructor for @State SomeObj's class.
102   = Given abstract content of generic type T.
103   //
104   public SomeObj (content) {
105     this.content = content;
106   }
107
108   @Override
109   protected T get() { // better
110     return this.content;
111   }
112
113   @Override
114   public MyObject filter(predicateCondition:()=> super T predicate) {
115     return (this.content == null && predicate.test(this.content)) ? MyObject.none() : this;
116   }
117 }

```

Helpsheet (MY)

E2 Helpsheet (MY)

Helpsheet (MY)

E2 Helpsheet (MY)