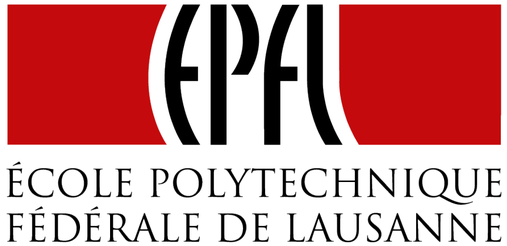


Subscriptions

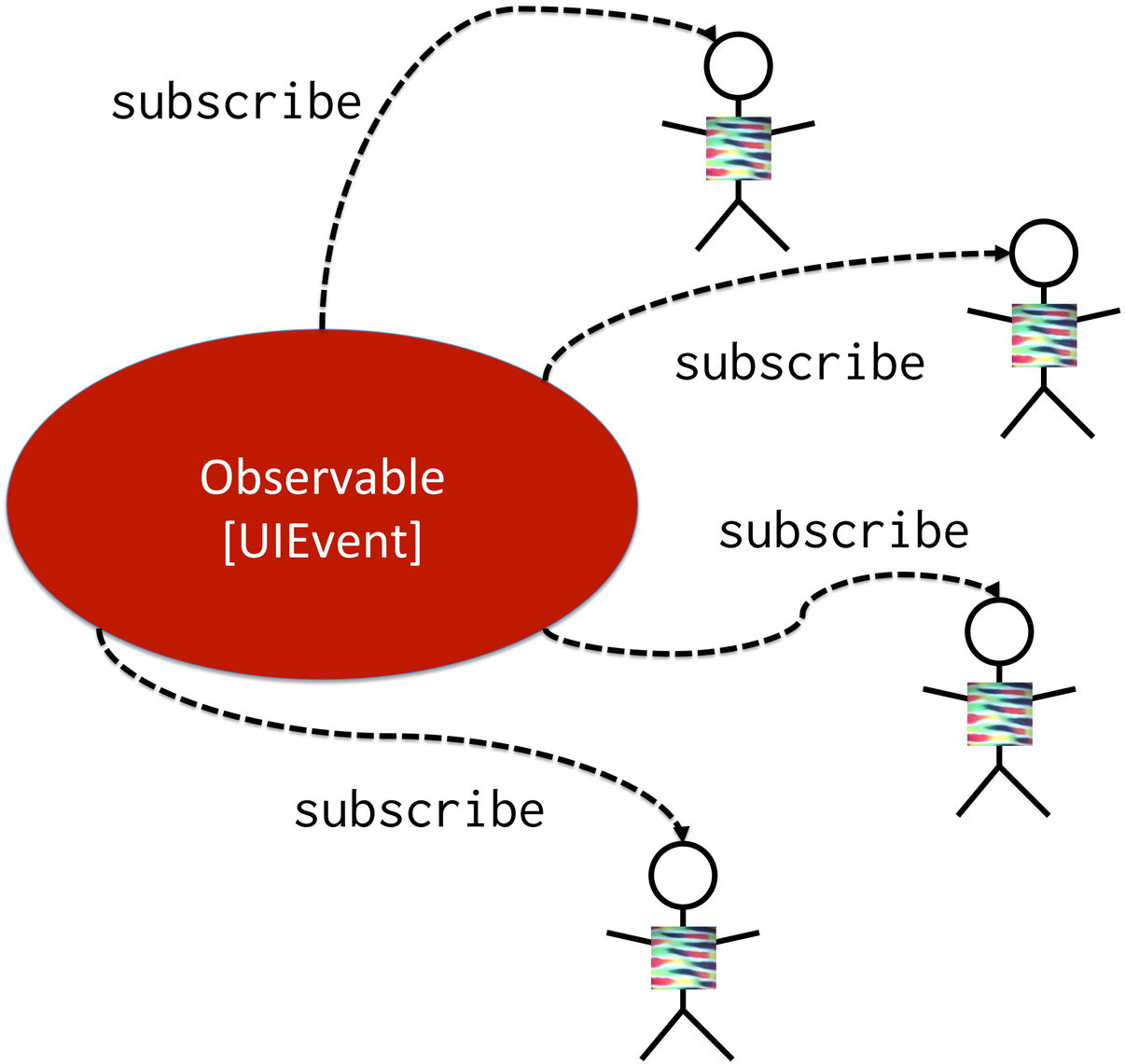
Principles of Reactive Programming

Erik Meijer

Unsubscribing from a stream

```
val quakes: Observable[EarthQuake] = …

val s: Subscription = quakes.Subscribe(…)

s.unsubscribe()
```

You're **Hot** and you're Cold



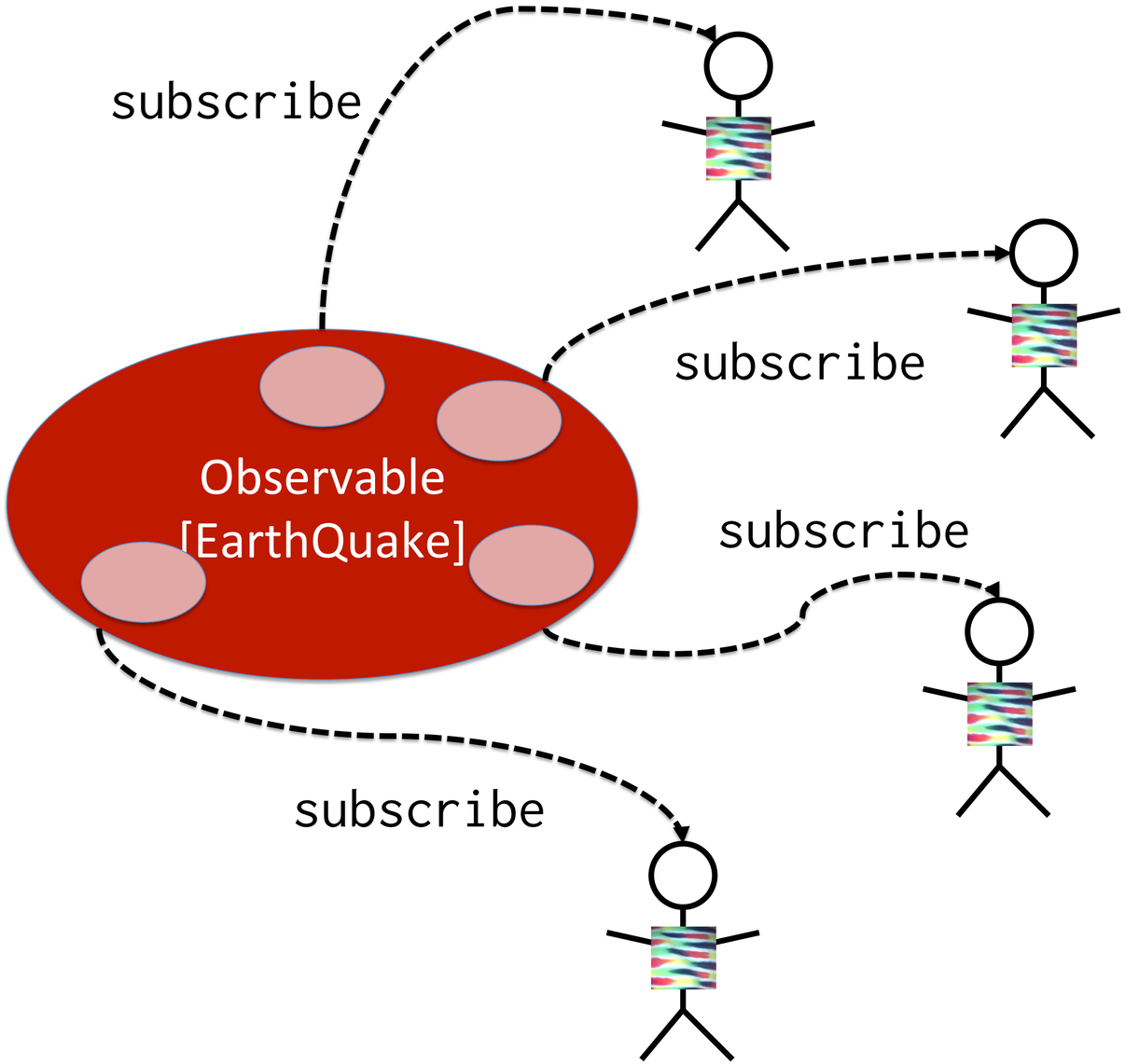
Hot Observable

≈

same source

shared by all subscribers

You’re Hot and you’re **Cold**

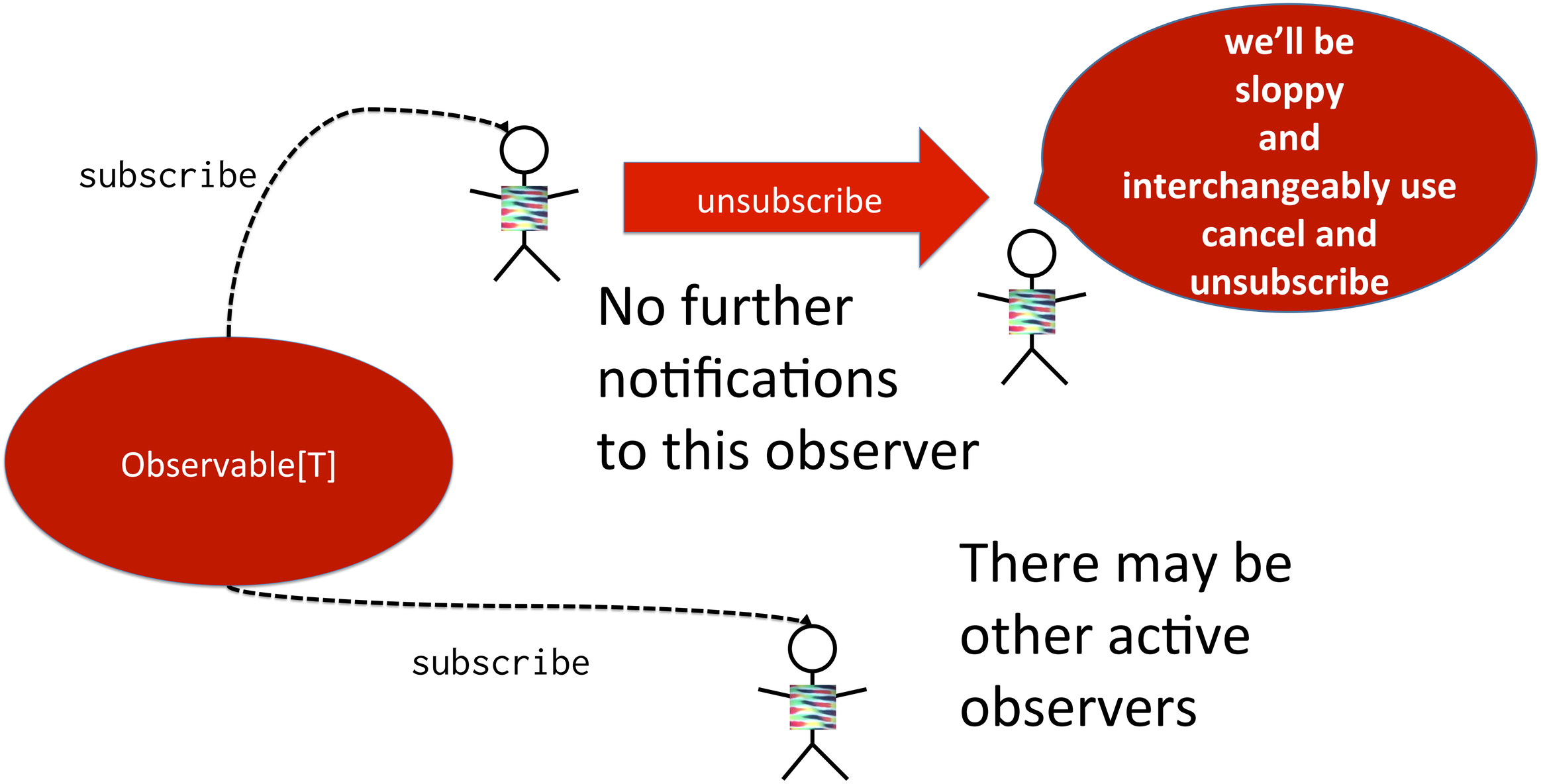


Cold Observable

≈

each subscriber
has its own private source

Unsubscribing != Cancellation



Subscriptions: The basics

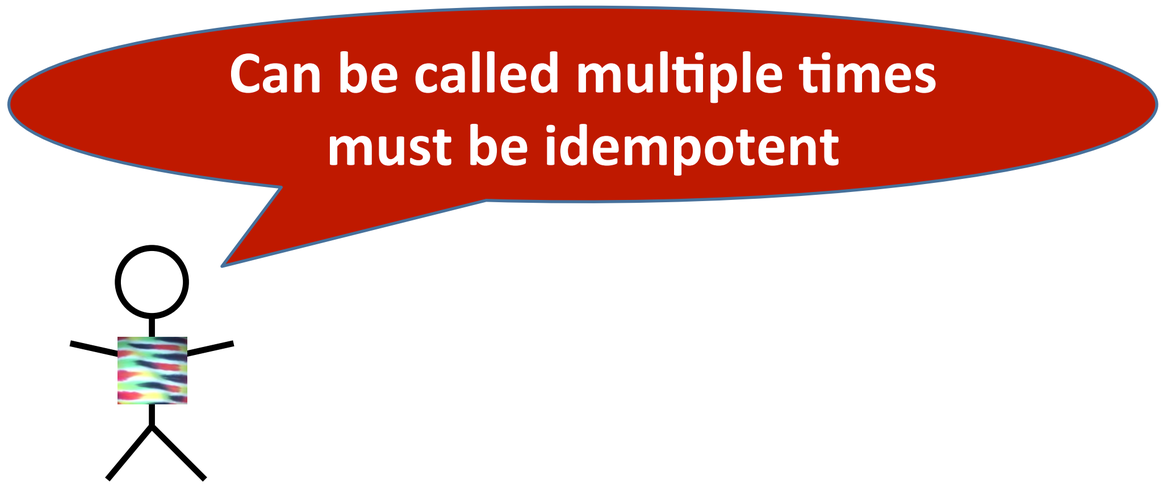
```
trait Subscription {
    def unsubscribe(): Unit
}

object Subscription {
  def apply(unsubscribe: ⇒ Unit):Subscription
}
```

Subscriptions: Meet the family

```
trait BooleanSubscription extends Subscription {
  def isUnsubscribed: Boolean
}

trait CompositeSubscription extends BooleanSubscription {
  def +=(s: Subscription): this.type
  def -=(s: Subscription): this.type
}

trait MultipleAssignmentSubscription extends BooleanSubscription {
  def subscription: Subscription
  def subscription_=(that: Subscription): this.type
}
```

Subscriptions

```
val subscription = Subscription {
    println("bye, bye, I'm out fishing")
}

subscription.unsubscribe()

subscription.unsubscribe()
```

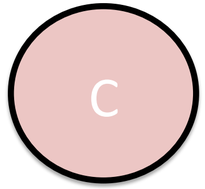


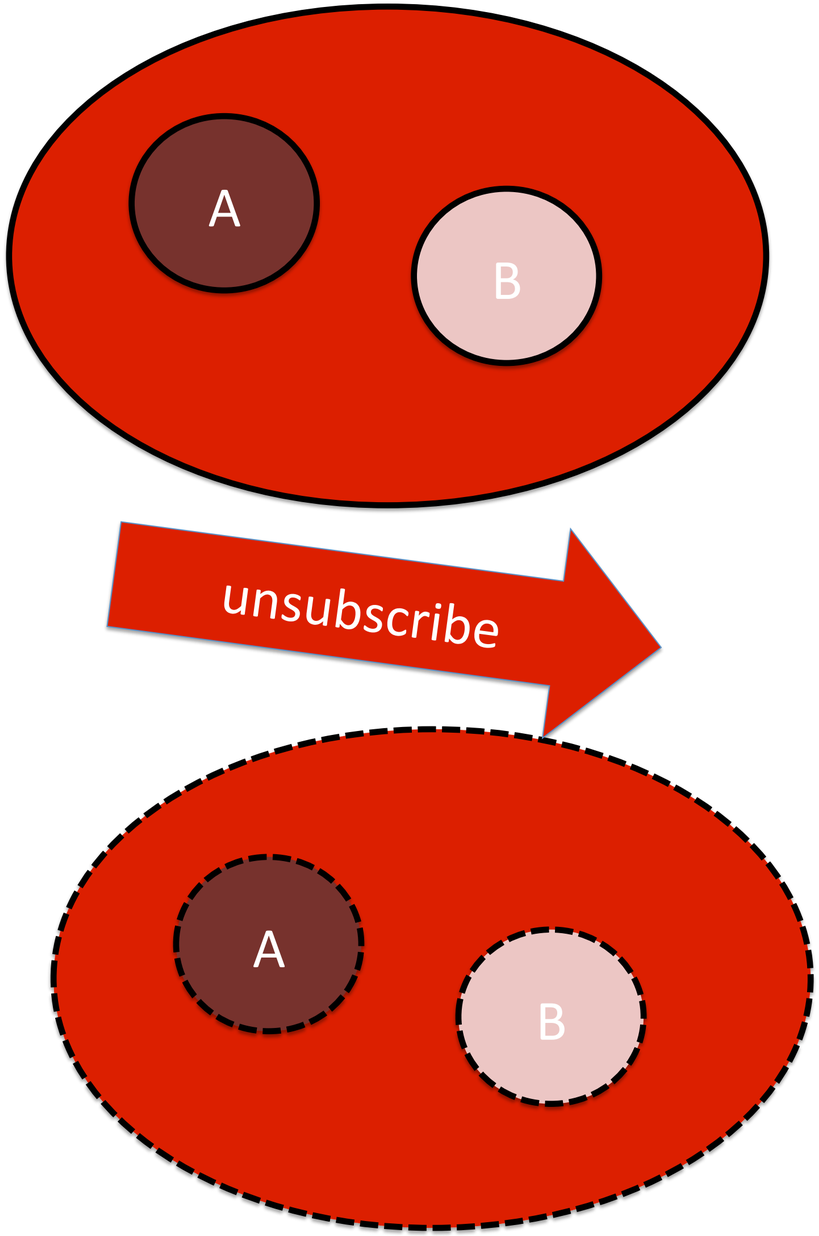
The Algebra of Subscription: BooleanSubscription

```
val subscription = BooleanSubscription {
   println("bye, bye, I'm out fishing")
}

println(subscription.isUnsubscribed)

subscription.unsubscribe()

println(subscription.isUnsubscribed)
```

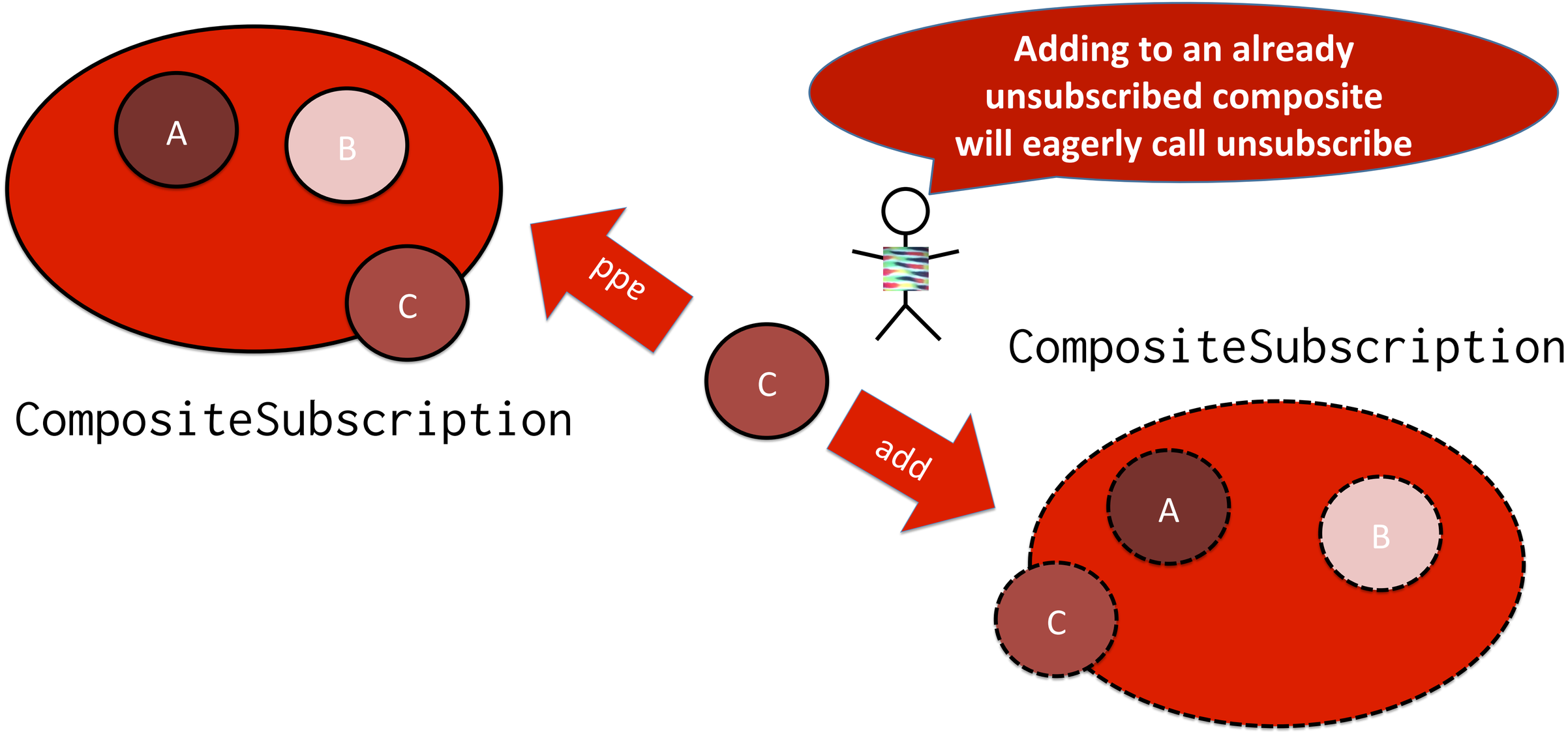
The Algebra of Subscription: CompositeSubscription

```
val a = BooleanSubscription { println(“A”) }
val b = Subscription { println(“B”) }

val composite = CompositeSubscription(a,b)

println(composite.isUnsubscribed)

composite.unsubscribe()

println(composite.isUnsubscribed)
println(a.isUnsubscribed)

composite += Subscription{ println (“C”) }
```



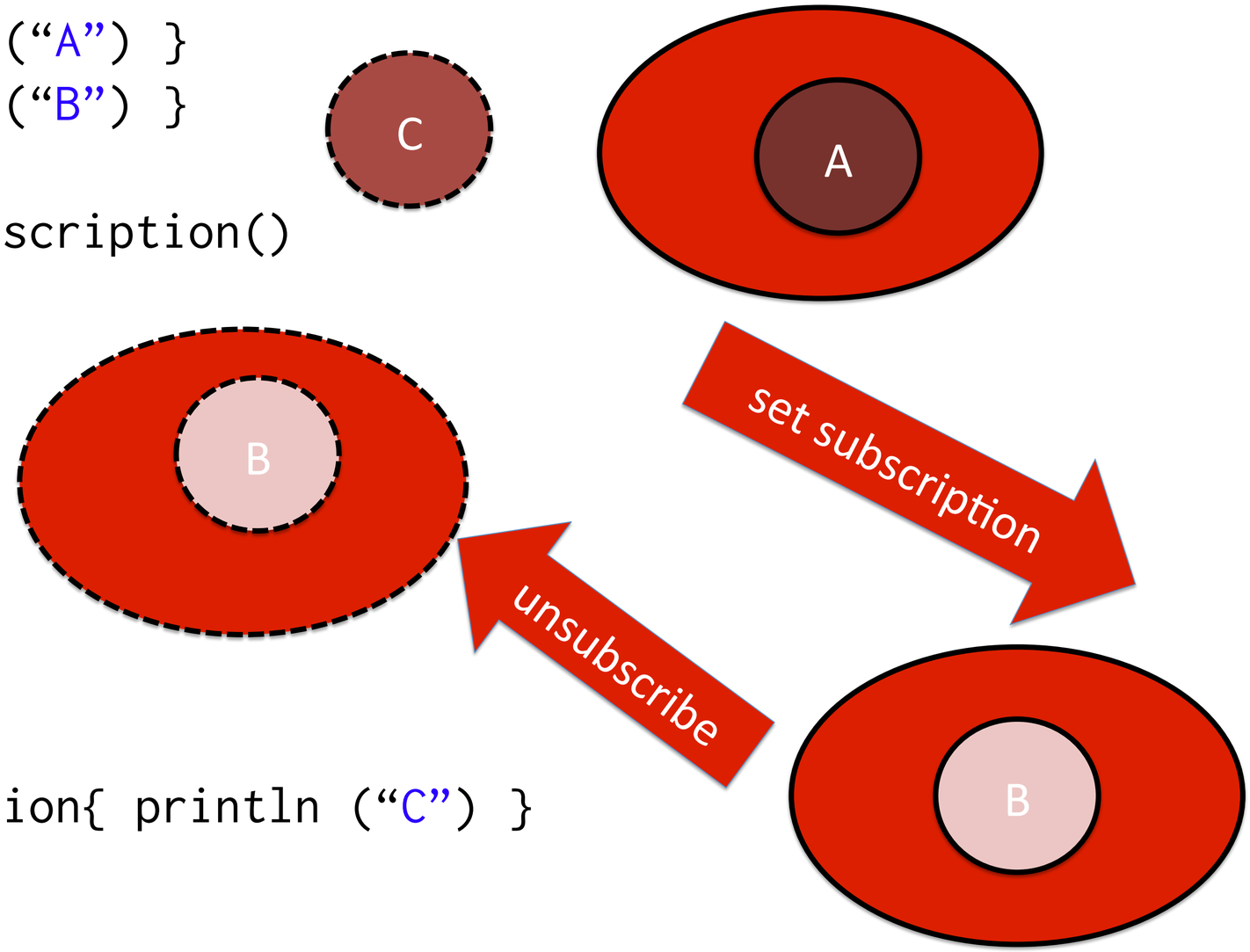


The Algebra of Subscription

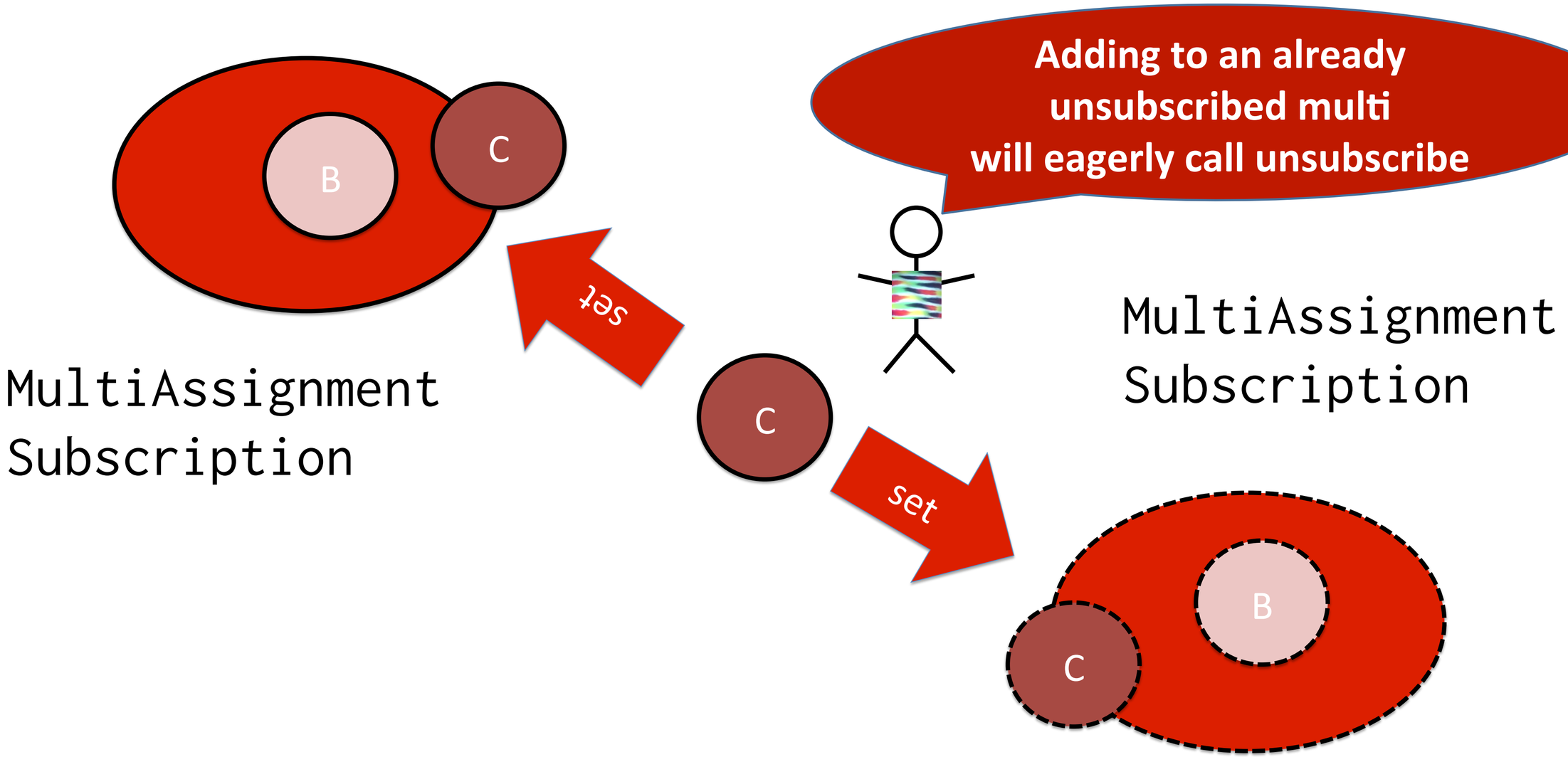


The Algebra of Subscription: MultiAssignment

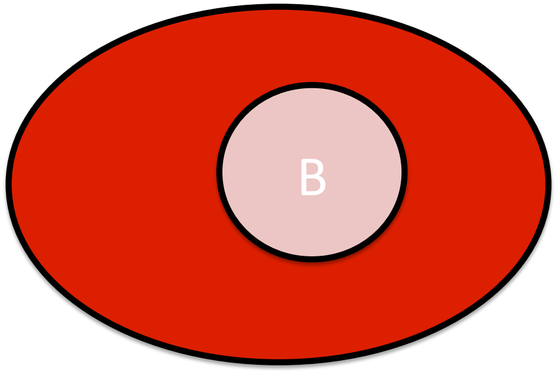
```
val a = Subscription { println(“A”) }
val b = Subscription { println(“B”) }

val multi = MultiAssignmentSubscription()

println(multi.isUnsubscribed)

multi.subscription = a
multi.subscription = b

multi.unsubscribe()

multi.subscription = Subscription{ println (“C”) }
```

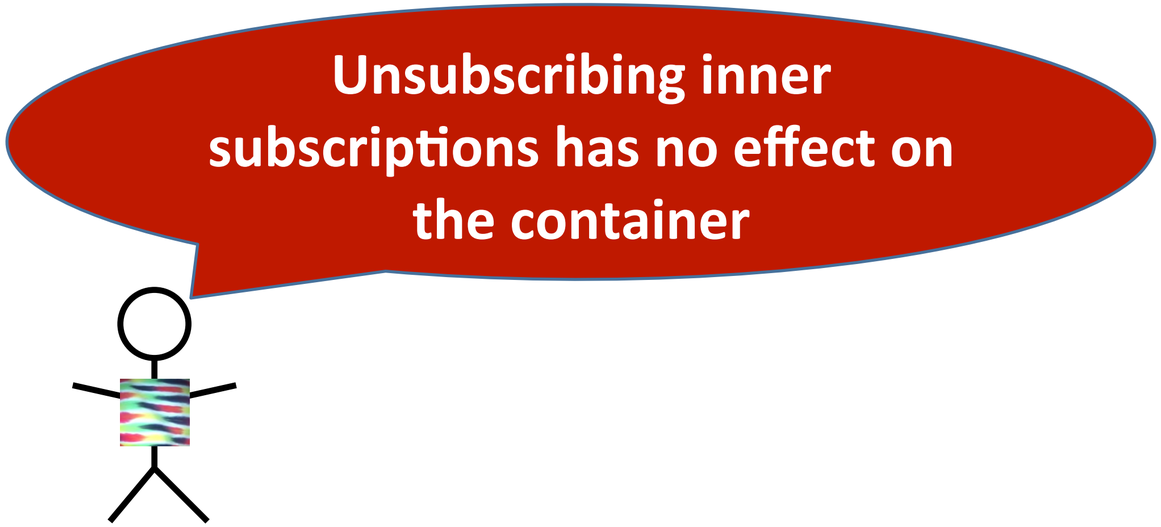


The Algebra of Subscription

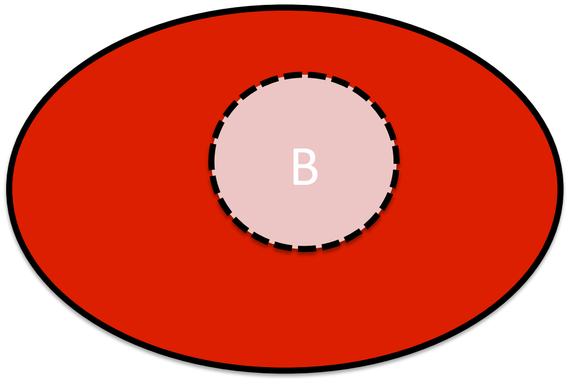


The Algebra of Subscription





`MultiAssignment/Composite Subscription`



Quiz

```
val a = BooleanSubscription { println(“A”) }
val b = Subscription { println(“B”) }
val c = CompositeSubscription(a,b)
val m = MultiAssignmentSubscription()
m.subscription = c
c.unsubscribe
```

a) b.isUnsubscribed == true

b) a.isUnsubscribed == false

c) m.isUnsubscribed == true

d) c.isUnsubscribed == true