



Reader Writer State

There 3 different kind of exposed variables



Read Variables

```
case class Config(feature: Boolean, maxSize: Int)

class Bar(config: Config) {

  def blah(x: Int): List[Int] =
    if(config.feature) List.fill(x min config.maxSize)(5)
    else Nil

  def foo(b: Boolean): Int =
    if(b) 10 else 5

}

val bar1 = new Bar(Config(true , 2))
val bar2 = new Bar(Config(false, 6))
```



Read Variables

```
case class Config(feature: Boolean, maxSize: Int)

class Bar(config: Config) {

  def blah(x: Int): List[Int] =
    if(config.feature) List.fill(x min config.maxSize)(5)
    else Nil

  def foo(b: Boolean): Int =
    if(b) 10 else 5

}

val bar1 = new Bar(Config(true , 2))
val bar2 = new Bar(Config(false, 6))
```

```
scala> bar1.blah(3)
res0: List[Int] = List(5, 5)

scala> bar2.blah(3)
res1: List[Int] = List()
```



Read Variables

```
def blah(config: Config, x: Int): List[Int] = {  
  if(config.feature)  
    List.fill(x min config.maxSize)(5)  
  else  
    Nil  
}  
  
def foo(b: Boolean): Int =  
  if(b) 10 else 5
```



Read Variables

```
def blah(config: Config, x: Int): List[Int] = {  
  if(config.feature)  
    List.fill(x min config.maxSize)(5)  
  else  
    Nil  
}  
  
def foo(b: Boolean): Int =  
  if(b) 10 else 5
```

```
scala> blah(Config(true , 2), 3)  
res2: List[Int] = List(5, 5)  
  
scala> blah(Config(false, 6), 3)  
res3: List[Int] = List()
```



Read Variables

```
class OrderRepository(dbConnection: Connection){  
  def listOrder: List[Order] = ...  
  def deleteOrder(id: Int): Unit = ...  
}
```



Read Variables

```
class OrderRepository(dbConnection: Connection){  
  def listOrder: List[Order] = ...  
  def deleteOrder(id: Int): Unit = ...  
}
```

```
object Main extends App {  
  val dbConnection = initialiseConnection  
  val orderRepository = new OrderRepository(dbConnection)  
  val userRepository = new UserRepository(dbConnection)  
}
```



Read Variables != Constants

```
val pi = 3.14  
  
def circleCircumference(radius: Double): Double =  
    2 * pi * radius
```



Read Variables != Constants

```
val pi = 3.14
```

```
def circleCircumference(radius: Double): Double =  
    2 * pi * radius
```

```
def circleCircumference(radius: Double): Double =  
    2 * pi * 3.14
```



Write Variables

```
val log: StringBuffer = new StringBuffer()

def doSomething(x: Int): Int = {
  log.append(s"called doSomething with $x\n")
  x % 2
}
```



Write Variables

```
val log: StringBuffer = new StringBuffer()

def doSomething(x: Int): Int = {
  log.append(s"called doSomething with $x\n")
  x % 2
}
```

```
scala> doSomething(5)
res4: Int = 1
```

```
scala> doSomething(6)
res5: Int = 0
```

```
scala> log
res6: StringBuffer =
called doSomething with 5
called doSomething with 6
```



Write Variables

```
var countCall: Int = 0

def isEven(x: Int): Boolean = {
  countCall += 1
  x % 2 == 0
}

def isOdd(x: Int): Boolean = {
  countCall += 1
  x % 2 == 1
}
```



Write Variables

```
var countCall: Int = 0

def isEven(x: Int): Boolean = {
  countCall += 1
  x % 2 == 0
}

def isOdd(x: Int): Boolean = {
  countCall += 1
  x % 2 == 1
}
```

```
scala> isEven(5)
res7: Boolean = false

scala> isEven(6)
res8: Boolean = true

scala> isOdd(6)
res9: Boolean = false

scala> countCall
res10: Int = 3
```



Write Variables

```
val five = isEven(5)  
val six  = isEven(6)
```

```
scala> countCall  
res11: Int = 2
```



Write Variables

```
val five = isEven(5)  
val six  = isEven(6)
```

```
scala> countCall  
res11: Int = 2
```

```
val five = isEven(5)  
val six  = !five
```

```
scala> countCall  
res12: Int = 1
```



Read / Write Variables

```
var count: Int = 0

def inc(n: Int): Int = {
  count += n
  count
}
```



Read / Write Variables

```
var count: Int = 0

def inc(n: Int): Int = {
  count += n
  count
}
```

```
scala> inc(5)
res13: Int = 5

scala> inc(2)
res14: Int = 7

scala> inc(5)
res15: Int = 12
```



Read / Write Variables

```
var seed: Int = 10

def genInt: Int = {
  seed = 7 * seed % 11
  seed
}
```



Read / Write Variables

```
var seed: Int = 10

def genInt: Int = {
  seed = 7 * seed % 11
  seed
}
```

```
scala> genInt
res16: Int = 4
```

```
scala> genInt
res17: Int = 6
```

```
scala> genInt
res18: Int = 9
```

