

3. Exemplary Solutions: A Simple Counter

Exercise 3.1: Accessor

You should put the accessor of `value` in the `accessing` protocol as clients are likely to need access to the value of a simple counter.

Exercise 3.2, 3.3, 3.4 are already explained in the exercise sheet.

Exercise 3.5: Adding an instance initialization method

Below is the code for `initialize`:

```
initialize
  "set the initial value of the value to 0"
  super initialize. "do not forget to invoke the super method!"
  value := 0.
```

Exercise 3.6 is trivial.

Exercise 3.7: SUnit

Test increment and decrement functionalities (Exercise 3.3):

```
testIncrementDecrement
  counter := SimpleCounter new.
  counter increment; increment.
  counter decrement.
  self assert: counter value = 1.
```

Test for Exercise 3.1 and 3.2:

```
testAccessors
  counter := SimpleCounter new.
  self assert: counter value = 0.
  counter value: 2
  self assert: counter value = 2.
```

Test for Exercise 3.4:

```
testPrinting
  counter := SimpleCounter new.
  self assert: counter printString
    = 'SimpleCounter with value: 0', String cr.
  counter increment; increment.
  self assert: counter printString
    = 'SimpleCounter with value: 2', String cr.
```

Test for Exercise 3.5:

```
testInitialize
  counter := SimpleCounter new.
  self assert: counter value = 0.
```

Exercise 3.8: Another instance creation method

First, write the test for creation method `withValue:`. Second, implement `withValue:` on the class-side of `SimpleCounter`.

```
testWithValueCreation
  counter := SimpleCounter withValue: 5.
  self assert: counter value = 5.
```

```
SimpleCounter class >> withValue: anInteger
  ^self new value: anInteger; yourself.
```

Exercise 3.9: Collections

```
| anArray sum |
sum := 0.
anArray := #(21 23 53 66 87).
anArray do: [:item | sum := sum + item].
sum
```

The evaluated value of this piece of code is 250.

Rewrite with indexing:

```
| anArray sum |
sum := 0.
anArray := #(21 23 53 66 87).
(1 to: anArray size) do: [:i | sum := sum + anArray at: i].
sum
```

Rewrite with `inject:into::`

```
| anArray |
anArray := #(21 23 53 66 87).
anArray inject: 0 into: [:sum :item | sum + item].
```
