10

Objects

Exercise 10.1. Modify the *make-counter* definition to add a *previous!* method that decrements the counter value by one.

Solution.

Exercise 10.2. [⋆] Define a *variable-counter* object that provides these methods:

make-variable-counter: Number → VariableCounter
Creates a *variable-counter* object with an initial counter value of 0 and an initial increment value given by the parameter.

set-increment!: Number → Void
Sets the increment amount for this counter to the input value.

next!: Void → Void
Adds the increment amount to the value of the counter.

get-count: Void → Number
Outputs the current value of the counter.

Here are some sample interactions using a *variable-counter* object:

```
> (define vcounter (make-variable-counter 1))
> (ask vcounter 'next!)
> (ask vcounter 'set-increment! 2)
> (ask vcounter 'next!)
> (ask vcounter 'get-count)
3
```

Solution.

Exercise 10.3. Define a *countdown* class that simulates a rocket launch countdown: it starts at some initial value, and counts down to zero, at which point the rocket is launched. Can you implement *countdown* as a subclass of *counter*?

Solution.

Exercise 10.4. Define the *variable-counter* object from Exercise 10.2 as a subclass of *counter*.

Solution.

Exercise 10.5. Define a new subclass of *parameterizable-counter* where the increment for each *next!* method application is a parameter to the constructor procedure. For example, (*make-parameterizable-counter* `0.1`) would produce a counter object whose counter has value `0.1` after one invocation of the *next!* method.

Solution.