```
Exercise 7-1
           Function basics
           Write a programm which simulates an EC2 instance.
            • The instance has a name and a state, which can be launched or stopped.
            • The Launch function should create a new instance and returns a pointer to a new instance.
            • The Stop function takes the instance pointer as parameter and sets the state to stopped.
            • Both functions return also errors, which the main function has to handle.
            • The main programm launches the instances and stops the same instance.
            • After each state change the instance name and status is printed.
                                                        Step 1: bash
                                mkdir instance && cd instance
                                                                Step 2: main.go
                    package main
                    import (
                         "fmt"
                    type InstanceState string
                    const (
                         InstanceStateRunning InstanceState = "running"
InstanceStateStopped InstanceState = "stopped"
                     Define instance states as string.
                     By defining an own type InstanceState we can ensure type safety.
                                                Step 3: main.go
                                    type Instance struct {
   Name string
                                         State InstanceState
                                     Define structure for the instance
                                                                            Step 4: main/main.go
func LaunchInstance(name string) (*Instance, error) {
    // Simulate lanching an instance by returning a new instance with state "running"
    return &Instance{Name: name, State: InstanceStateRunning}, nil
                         The parameter name sets the instance name of a new struct.
                                                                    Step 5: main.go
                       func StopInstance(instance *Instance) error {
   instance.State = InstanceStateStopped
                            return nil
               The parameter is a pointer. So when we change data, it will change the original data.
                                                                          Step 6: main.go
          func Observe(instance *Instance) {
              fmt.Printf("Instance %s is %v\n", instance.Name, instance.State)
                         Define observe.
                         Now we have all data and function. Starting do declare main.
                                                                      Step 7: main.go
              instance, err := LaunchInstance("Alice")
              if err != nil {
                   fmt.Printf("Error starting instance: %s\n", err.Error())
                   return
                                 Call Launch and handle error in main().
                                                                      Step 8: main.go
              instance, err := LaunchInstance("Alice")
              if err != nil {
                   fmt.Printf("Error starting instance: %s\n", err.Error())
                   return
              Observe(instance)
                                 Calling Observe for instance status
                                                                      Step 9: main.go
              err = StopInstance(instance)
              if err != nil {
                   fmt.Printf("Error stopping instance: %s\n", err.Error())
                   return
              Observe(instance)
                                            StopInstance
                                                 Step 10: Output
```

```
3 Define structure for the instance4 define a LaunchInstance function,
```

Create an empty root/work directory

Define instance states as string

Instance Alice is running Instance Alice is stopped

define a LaunchInstance function, return Instance*Define StopInstance

Call Launch and handle error.

Output of the called funtion

6 Define observe function

2

7

10

8 Call Observer for instance status
9 Call StopInstance