Spec:

The Test

Write a simple Zoo simulator which contains 3 different types of animal: monkey, giraffe and elephant. The zoo should open with 5 of each type of animal.

Each animal has a health value held as a percentage (100% is completely healthy). Every animal starts
at 100% health. This value should be a floating point value.

The application should act as a simulator, with time passing at the rate of 1 hour every 20 seconds.

Every hour that passes, a random value between 0 and 20 is to be generated for each animal. This
value should be passed to the appropriate animal, whose health is then reduced **by that percentage of their current health.**

The user must be able to feed the animals in the zoo. When this happens, the zoo should generate
three random values between 10 and 25; one for each type of animal. The health of the respective
animals is to be increased by the specified percentage of their current health. Health should be capped at 100%.

When an Elephant has a health below 70% it cannot walk. **If its health does not return above 70% once the subsequent hour has elapsed, it is pronounced dead.**

When a Monkey has a health below 30%, or a Giraffe below 50%, it is pronounced dead straight away.

The user interface should show the current status of each Animal and contain a single button to feed the zoo. The UI should update to reflect each change in state, and the current time at the zoo.

Design

Testing

Manual test:

Start the application,

Check that there are 5 of each type of animal, they show their state and health %

Check the clock is shown and increments by 3 minutes every second.

Check the animals' health decreases every 20 seconds of real world time (or hour of zoo time)

Check the animals don't decrease their health by the same amount or more than 20%

Check the elephants are shown as unable to walk when they fall below 70%

Check they die the next time their health falls.

Check that monkeys die when they fall below 30% and giraffes when below 50%

Check that the feed button heals the animals

Check that you can save elephants that can't walk by feeding them

Automated test

These should be added time permitting.

UI Layer

Display

- animal data to the user
  show
  - Animal type
  - Health
  - State
- Feed button
- Time at the zoo

Use a winform because this is a demo and win forms are quick.
Use a difference namespace to the service layer call it ZooSimulator.InterfaceLayer
The form should hold a reference to an object that fulfills the zoo interface.
The form should not contain and business logic
When the feed button is pressed pass the call directly to the zoo object.
Poll the zoo object every second for animal data and the time.
Expect to be passed an list of a struct containing:

- Animal unique id
- Animal Type
- Animal state
- Animal health

Service Layer

Contain all the service layer in a single namespace, call it ZooCorp.ZooDemo.ServiceLayer
Use dependency injection via autofac. Do not create any objects outside of factories other than via DI.

Classes

Struct AnimalData

Members

- Unique id
- Health
- Category enum (instant death or danger zone)
- Type (elephant etc)
- State

Interface IZoo

Method to feed all the animals, no arguments

Method to return the current time at the zoo and an list of animal data structs

Class Zoo :IZoo

Members

- IDatabaseHandler
- List of IAnimal Animals
- Timer
- Time TimeAtTheZoo

Methods

Implement IZoo

```
Public FeedAllAnimals()
    Loop over Animals
    generate
    three random values between 10 and 25; one for each type of animal. The health of the
    respective animals is to be increased by the specified percentage of their current
    health. Health = health * (1+%). Health should be capped at 100%.
    Apply changes to database
```

```
Private OnHourEvent()
    Loop over Animals
    For each animal
        Call Animal.OnHourEvent()
        Generate a random number between 0 and 20
        Animal.health = animal.health * (1 - the random number)
    Apply changes to database
```

```
On creation
    Ask the DatabaseHandler for the current animals.
    Ask the DatabaseHandler for the damage over time duration
    Start the timer.
```

Using the timer, increment TimeAtTheZoo by 3 minutes every second. Every hour on the hour call OnHourEvent().

IAnimal

Unique id property int

Health property Float

State property enum

Void OnHourEvent()

AnimalData CompileData()

Abstract Class Animal :IAnimal

```
OnHourEvent()
{
 // Do nothing
}

CompileData()
Make a struct and populate with own data, return it.
```

InstantDeathAnimal : IAnimal, Animal

```
(Monkey and giraffe)
Private property int DeathThreshold, set on creation
Override the Heath setter and if health < DeathThreshold change state to dead.
Disallow changes to health if dead.
```

DangerZoneAnimal : IAnimal, Animal

```
(Elephant)
Public property DangerZoneCustomerFacingName
Private property int DangerZoneThreshold, set on creation
Override the Heath setter and if health < DangerZoneThreshold change state to DangerZone.
If health is increased above DangerZoneThreshold, change state to healthy.
Disallow changes to health if dead.
Override OnHourEvent()
    Check if health below DangerZoneThreshold and if so change state to dead.
```

DatabaseHandler (wrap in an interface)

```
Members
Private IDatabaseService
Private IAnimalFactory

Methods
Public FetchAnimals()
    Call DataBaseService.FetchAnimals()
    Pass results to IAnimalFactory
    Return results from animal factory.

Public UpdateDatabase (animals list)
    Convert Animals into an list of animal data structs
    Pass to DatabaseService
```

DatabaseService (Wrap in an interface)

Methods

Fetch animals()

Build and return an list of animal data structs to meet the spec

“3 different types of animal: monkey, giraffe and elephant. The zoo should open with 5 of each type of animal.”

```
UpdateDatabase(list of animal data structs)
 {
        // Demo app so do nothing
}
```

Animal Factory (wrap in an interface)

MakeAnimal(animal data struct)

Make a new animal of the correct Category.

Stretch goals:

Pull in the feed and damage % from the DB.
Add logging of feed and death events.
Add ability to feed 1 animal in particular.
Move the healing and damage code into the animal object?
Give the animals names