Exercise: People Counter

Goal

Form pairs and discuss how you can refactor the code below. Make notes on the different steps you would take to do this in a safe way.

Domain

For safety, we need to make sure we don't exceed the maximum occupancy in any zone of the building.

We've installed cameras pointed at all the entrances of each zone of the building. The cameras can count people coming in and going out. We can query this data using the cameras' API. We calculate the occupancy of each zone and show it on a monitor in the security's control room.

Tips

Some questions you can ask yourselves:

- Can we epxress the domain language better? What domain concepts are hidden or implicit?
- Are there any (DDD) design patterns we can use? Can we make use of immutability? Are there opportunities for introducing better abstractions?
- How can we test this code (or make it more testable)?
- Can we isolate side effects? Can we introduce Hexagonal Architecture (aka Ports & Adapters)?

Advanced: - Rewrite this in a FP language (or using FP style in an OOP language). - Find opportunities to use monoids, map/fold/filter/...

Code

```
+ main()
    httpClient = new HttpClient()
    lobby = new Zone('South Auditorium',
        [ new Counter(httpClient, 'Camera1', '192.168.55.130')
        , new Counter(httpClient, 'Camera2', '192.168.55.131')
        , new Counter(httpClient, 'Camera3', '192.168.55.132')
        , new Counter(httpClient, 'Camera4', '192.168.55.133')
        , new Counter(httpClient, 'Camera5', '192.168.55.134')
        ])

    repeat
        lobby.update()
        print "Lobby: {lobby.occupancy()}"
        sleep(1)
    endrepeat
```

```
class Counter
    - ip
    - name
    - total_in = 0
    - total_out = 0
    - friendly_name = ""
    - last_udpate
    - serial
    - httpClient

    + constructor(httpClient, ip, name)
        this.httpClient = httpClient
        this.ip = ip
        this.name = name
        update()

    + update()
        try
            data = httpClient.get("http://{ip}/people-counter/api/live.json")
            total_in = data['in']
            total_out = data['out']
            friendly_name = data['name']
            last_udpate = data['timestamp']
            serial = data['serial']
        catch (Exception e)
            friendly_name = "error connecting to {name}"


    + totalIn()
        return total_in

    + totalOut()
        return total_out

    + friendlyName()
        return friendly_name
```

```
class Zone
    name
    counters = []
    totalIn = 0
    totalOut = 0
    occupancy = 0

    constructor(name, counters)
    {
        this.name = name
        this.counters = counters
        this.update()
    }

    + update()
        totalIn = 0
        totalOut = 0
        for (counter in counters)
            update()
            totalIn += counter.totalIn()
            totalOut += counter.totalOut()
        }
        occupancy = totalIn - totalOut
    }

    + counters()
        return counters

    + totalIn()
        return totalIn

    + totalOut()
        return totalOut

    + occupancy()
        return occupancy
```