Tutorial 7: Streams

CSE1100 - Object Oriented Programming

1 Rewriting with Lambdas

You are given a writer application. It asks the user for input and then uses a thread to write to file what the user put in. The `WriteTask` is responsible for writing. Since we only use `WriteTask` at this one point in the application and do not plan to reuse it, we can rewrite to not have a separate class.

1.1 Anonymous class

Rewrite the `WriterApplication` to use an anonymous class. You can use this reference.

1.2 Lambda

The anonymous class still looks bulky and there is a lot of syntax to write for a small amount of functionality. Now rewrite the anonymous class to a lambda.

2 Streams

Suppose you want to go do something in Delft with your newly found friends. Being a computer scientist, you programme a `VenueCatalog` that stores `Venue`s, which are either `Bar`s or `Restaurant`s. Now you just need to decide where to go.

2.1 A quality venue

Because you do not know Delft yet, you let the reviews guide you. Implement the `qualityVenues` method in `VenueCatalogue`. It should return the list of `Venue`s with 3 stars or more. Make use of `Stream` and `filter`.

2.2 Pleasing everyone

You just found out one of your friends is vegan and you do not want them to feel left out. Implement the `veganRestaurantNames` method in `VenueCatalogue`. It should return the names of all the vegan `Restaurant`s. Make use of `Stream`, `filter` and `map`.

2.3 Cheap drinks

Your last visit to the vegan restaurant left a bit of a dent in your wallet, so you decide to just grab drinks. Of course you want cheap drinks. Implement the `cheapDrinkingLocations` method in `VenueCatalogue`. It should return the locations of all bars where a beer costs less than 2 euros. Make use of `Stream`, `filter` and `map`.

2.4 Testing

Write tests to verify each of the methods you wrote works.

3 Optionals

You are given a `CalculatorApplication`. It has a method `safeDivide`. This method takes two doubles, a divisor and a dividend and performs a division. However, as you know, dividing by zero is not allowed. In this case the method should return an empty optional. Implement the `safeDivide` method.

4 Threads

We have created a `DataProcessingApplication` that reads 10,000,000 numbers from `numbers.txt` and sums them. As this can take some time, we print a progress bar every time we read a number, but this makes the programme slow. Using threads, make the progress bar print in parallel every 10 milliseconds. You are allowed to change anything (including moving methods into different classes or changing access modifiers). When implemented correctly, the programme should finish in a few seconds rather than more than a minute.

5 Synchronisation

We have created an implementation of a queuing system. Students can ask questions in the form of a `Request`, which then will get added to the `Queue`. There is one problem however: when two assistants call `getNext` at the same time, they sometimes will receive the same `Request`. We also want to make sure `enqueue` and `getNext` always work in the order they are called.

Change the `Queue` class such that the `enqueue` and `getNext` methods work as we want them to, regardless of the thread that is calling them and the time it takes to add and remove requests from the queue.

Testing multithreaded behaviour is almost impossible, so you will not have to write any tests. There are already tests provided in the `QueueSyncTest` class, note that you do **not** have to understand these tests.

6 Client/Server

This topic is retired, and therefore not part of the exam material

The local library has created a new server. Unfortunately, this new architecture requires that their client be rewritten. You are tasked with writing a new `LibraryClient`. The server is already implemented and handles most of the logic. The client should connect to the server, write all messages from the server to console, and send all user input to the server.