SED : Exercise 8 : System Integration - Work in Pairs

In this tutorial we look at patterns that can help us to work with third party code, and integrate existing components into our applications, whilst maintaining a good design for our own code.

Look at pages 2 onwards of this spec for details of how to get the skeleton code, test it, and submit.

Imagine you are building an application that uses temperature data (maybe it is a news app, a holiday destination finder, or something similar). You have a library that can retrieve weather forecasts for you, but accessing this data can be slow. Your task is to build a local cache for this data to be used by your application, so that data can be returned more quickly - we assume that the weather doesn't change that often.

Weather Client

You are given a 3rd party client that can connect to a remote weather service (a fictional weather.com) and retrieve data given a "region" and a day. The jar file weather.jar contains the given weather client. Here is an example of how to use the client:

```
import com.weather.*;

Forecaster forecaster = new Forecaster();
Forecast londonForecast = forecaster.forecastFor(Region.LONDON, Day.MONDAY);

System.out.println("London outlook: " + londonForecast.summary());
System.out.println("London temperature: " + londonForecast.temperature());
```

Exercise

Build your own client, that uses the given Forecaster to retrieve data, but caches it in memory to speed up subsequent lookups.

It should be possible to set a limit for the size of the cache, evicting old entries if the maximum size is reached.

Extension: Data should be cached for up to one hour, after which time it should be refreshed.

Questions to Consider

- Have you tested the behaviour of your cache effectively? Think about unit vs integration tests.
- Does your API follow the design of the weather.com API? Does it have to? What API would be best for *your* application?
- If the third party changed their API, how much of your code would be affected?

Getting The Skeleton Code

Get the outline from GitLab:

```
git clone https://gitlab.doc.ic.ac.uk/lab2122_spring/sed_ex8_login.git
```

Project Structure

When you clone from GitLab, you should find a similar structure to what we had in previous exercises. There is an Example.java showing how to call the third party API. You are free to create whatever classes you need for the exercise.

```
├── lib
│   └── weather.jar
├── build.gradle
├── build.sh
└── src
    ├── main
    │   └── java
    │       └── ic
    │           └── doc
    │               ├── Example.java
    │               └── README.txt
    └── test
        └── java
            └── ic
                └── doc
                    └── README.txt
```

Do follow the existing structure of directories and packages for code and tests, as per previous weeks. Delete the Example.java and README.txt files once you have added your code.

Running the Build

This is the same as previous weeks, you can just type `./build.sh`

Make sure you run `./build.sh` before you submit, as this is what the autotest will run.

If you have completed the required functionality, the build passes, and you are happy with your code then you are ready to submit.

Submission

When you have finished, make sure you have pushed all your changes to GitLab (source code only, not generated files under target), test on LabTS and submit your token to CATE.

Deadline

There is a lab session timetabled for Tuesday 14:00-16:00 UK time where you can talk to us about the exercise. This is not intended to be a large exercise, so it should not take a lot of time.

The deadline for submission to CATE is **7pm UK time on Thursday 10th March**.

Only one person (whoever cloned the repository originally) needs to submit from LabTS to CATE. Once this is done, add your partner as a group member on CATE. The other person should sign the submission on CATE to confirm.

Assessment

The markers expect that your submission passes the automated tests and checks:

- Code compiles
- Tests pass
- Style and test coverage checks pass

Make sure you have 3/3 on LabTS.

If you pass these automated checks then the markers will review the design of your code and award marks based on:

- Code meets requirements / effective implementation of the cache
- Effective testing
- General code quality / freedom from duplication etc
- (Extension) effective implementation of expiring cached items after 1 hour
- Bonus marks for particularly good code or design (at the marker's discretion)