Expeditors Backend Academy Labs

Introduction

This document contains the labs for the Expeditors Backend Academy.

The instructions are split up into **Classwork** and **Homework**. The Classwork we will do in class together. You should do the Homework in the **Homework** module. Create a new package for each week's homework if it makes sense. I will go through an example before we start.

Week 4

Classwork

1. Designing a Service. Now we are going to start using our Student class for something useful. We need some way for users to interact with your Students, i.e. we need to create a StudentService. First step is going to be coming up with a rough design. We are going to do this together.
2. The StudentService should, at the very least, allow users to

    a. Add, Delete and Update a Student
    b. Get a Student by id
    c. Get all Students

3. At this stage, we are not going to use a database. All data will be kept in memory. But you should keep in mind that eventually you will be using a real database. Things to think about:

    a. How should the functionality should be partitioned – what are the different *kinds* of work you have to do.
    b. This will lead you to the different classes you will need, and what their responsibilities should be.
    Hint – the *Single Responsibility Principle* is OO speak for saying a class should have responsibility over only one part of the functionality of an application. Or, as Robert Martin puts it "a class should have only one reason to change".
4. What kind of data structures should be used.

5. How can we organize our code so that it will allow us to painlessly switch to using a real database at some point in the future. Designing for change.

Homework

Objectives

1. Design and write an application consisting of many moving parts.
2. Use OOP techniques to create classes which can be substituted for each other

Tasks

1. Design a AdoptionService along the lines of the StudentService we implemented in class. Use all the tricks we learned today.
2. Create a second DAO (also storing data in memory for now). It can even be just a copy of the one you have, obviously with a different class name.
3. Set up your code so that you can easily switch between the DAOs for any particular run of your Application.
   a. Hint - think Factory pattern.
4. Unit tests!!

Week 5

Classwork

1. Fun and games with lambdas and streams

Homework

Objectives

1. Use lambdas and streams
2. Use simple functional programming techniques

Tasks

1. Add functionality to your service to retrieve Adopters by name.
2. Make your Adopter class sortable by “natural order”.
3. Try and sort your Adopters by some other criteria, e.g. by date of adoption
4. Optional -- Write a method called findBy, which will allow you to search for Adopters by some user supplied criteria.