Exercise 3: Liskov Substitution Principle

TECHNISCHE UNIVERSITÄT DARMSTADT

Software Engineering Design & Construction
SS 2015 - Dr. Michael Eichberg

The purpose of this exercise is to get familiar with the Liskov Substitution Principle (LSP), in particular how it manifests itself in types and variances[1].
Your solution should be as simple as possible. As a first step, make yourself familiar with the code. Add your implementations in the corresponding places and test them thoroughly. As a minimum requirement, your code must at least compile without errors.

Introduction

You will implement a small class hierarchy for immutable tuples in Scala and Java. A tuple is a sequence of elements of a fixed length. The base trait/fully abstract class `Tuple` should take one type parameter `A` that denotes the type of elements the tuple can contain. Every tuple of length n should expose the following methods:

- A `length` method returning the number of elements n in the tuple.
- A `get` method, returning the element at a given index (starting at 0). It throws an exception if the index is < 0 or $\geq n$.
- A `contains` method that checks whether a given object is an element of the tuple.
- An `add` method that creates a new tuple of length $n+1$ that contains the elements of the existing tuple with a given element appended.
- A `map` method that executes a function on each element of the tuple.

You should implement concrete subclasses `Tuple0`, `Singleton`, `Pair` and `TupleN` for tuples of length 0, 1, 2 or any number ≥ 0, respectively. Their constructors should take the corresponding number of elements. For `TupleN`, you are free to use varargs, an array or some collection.

Task 1 Scala

Implement the 5 tuple classes and 5 methods in Scala. First, **use variance annotations** `+` or `-` for every type parameter for which they are possible. Second, the **types of the methods should be as precise as possible**. Third, please implement the `get` method as `apply` method. In particular, the `contains` method should have a parameter with a type more precise than `Any` or `Object`.
Can you define `Tuple0` as a Scala **`object`** instead of a Scala **`class`**? If you can, simply replace the class definition with a corresponding object definition. If you cannot, leave a comment why.
According to the Liskov Substitution Principle, could any of the concrete tuple classes inherit from each other? If yes, leave a short comment how, and why another way is not possible. If you don't think it is possible, also leave a comment why. (Please don't try to implement it, though).

Task 2 Java

Implement the 5 tuple classes and 5 methods in Java. Since Java has use site variance, you cannot use variance annotations. However, again, the **types of the methods should be as precise as possible**. They will be different (potentially less precise) from the Scala solution, though, but the `add` method should not contain imprecise types such as `Object`. Hint: you are free to make the `add` method static in order to achieve this.

[1] It is worth to have a look at the Wikipedia pages at `http://en.wikipedia.org/wiki/Liskov_substitution_principle` and `http://en.wikipedia.org/wiki/Covariance_and_contravariance_(computer_science)`

Information mid-term exam

On the 22 of May, 11:40 a.m. we write the mid-term exam. The room splitting is as follows:

- People with the starting letter A - J write the exam in room S202/C205.
- People with letters from K - Z write in room S105/122.

Information reference solution

The reference solution for this exercise is published not on the 22th of May. Instead we publish them on the 29th of May and discuss it previously in the exercise.
As I have heard no one handed-in a solution for the previous exercises, please take your chance to get feedback on your implementation, because only then we could help you.

Solve it on your own!

Although this exercise is not graded, it is highly recommended to do it by yourself. Just looking at a solution is much easier in comparison to actually coming up with it.

Requirements if you want to submit your solution

You can, once in the semester, submit your solution to get it corrected. Send your solution to `weiel@st.informatik.tu-darmstadt.de`. Make sure you zip your complete sbt project and make sure that is out of the box working by running `sbt run` and `sbt test`. If the project doesn't compile properly you will not receive any feedback!