Software Engineering

WS 2021/22, Assignment 02

Prof. Dr. Sven Apel
Annabelle Bergum
Sebastian Böhm
Christian Hechtl

Handout: 23.11.2021
Handin: 07.12.2021 23:59 CET

Organizational Section:

- The assignment must be accomplished by yourself. You are not allowed to collaborate with anyone. Plagiarism leads to failing the assignment.
- The deadline for the submission is fixed. A late submission leads to a desk reject of the assignment.
- We provide a project skeleton that must be used for the assignment. The skeleton can be downloaded from the CMS.
- The submission must consist of a *ZIP* archive containing only the project folder (i.e., the folder included in the provided skeleton)
- Questions regarding the assignment can be asked in the forum or during tutorial sessions. Please don't share any parts that are specific to your solution, as we will have to count that as attempted plagiarism.
- If you encounter any technical issues, inform us as early as possible.

Task 1 [30 Points]

Your task is to implement a configurable collection (think lists, sets, heaps, ...) in Scala[1] using different implementation techniques. We provide an implementation using C preprocessor directives which acts as a template for the other implementations. We highly recommend to solve the three tasks in order since this gives you a more gentle introduction to Scala before using more advanced concepts. If you are unfamiliar with the language we recommend taking the *tour of Scala*[2].

a) Implement the configurable collection using **runtime parameters** in the file `runtime/Collection.scala`. We provide the collection interface and a configuration class that gets passed to the constructor of the collection class. Your implementation must change its behavior based on the values in the configuration object.[10 Points]

b) Implement the configurable collection using the **decorator pattern** in the file `decorator/Collection.scala`. We provide the component interface as well as tests that specify how the concrete component and decorator classes must be named and how they can be combined. [10 Points]

c) Implement the configurable collection using **traits**[3] **and class composition with mixins**[4] in the file `traits/Collection.scala`. We provide the collection's interface as well as tests that specify how the traits must be named and how they can be combined. [10 Points]

Grading Your submission will be graded based on the following criteria:

- Your submission must be in the correct format, i.e., a *ZIP* archive with the same layout as the project skeleton (may result in 0 points if violated).
- Your submission must compile, i.e., the command `sbt compile` must succeed (may result in 0 points if violated).
- Each subtask must be implemented using the specified implementation technique (may result in 0 points for the subtask if violated).
- We run unit tests (the ones provided + additional tests) against your submission. Points will be deducted for each failing test.

[1] `https://www.scala-lang.org/`
[2] `https://docs.scala-lang.org/tour/tour-of-scala.html`
[3] `https://docs.scala-lang.org/tour/traits.html`
[4] `https://docs.scala-lang.org/tour/mixin-class-composition.html`

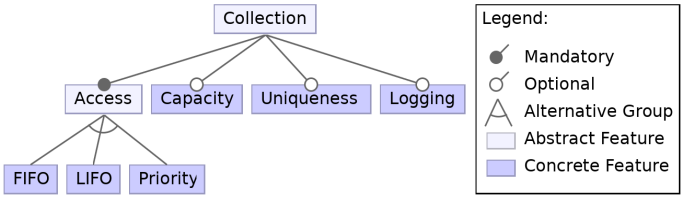
Project Skeleton You must implement your solution based on the provided project skeleton. The skeleton has the following structure:

```
CollectionPreprocessor.scala.txt
build.sbt
src
  main
    scala
      decorator
        Collection.scala
      runtime
        Collection.scala
      traits
        Collection.scala
  tests
    scala
      *CollectionTest.scala
```

The file `CollectionPreprocessor.scala.txt` contains a (naive and inefficient) implementation of the configurable collection using C preprocessor directives. We also provide some interfaces and tests for each task which can be found in the respective subfolders of `src`. The file `build.sbt` contains a build script that we use to build your submission and run the tests.[5] You must not edit this file.

The build script should also enable you to import the project skeleton into any IDE with Scala support (e.g., IntelliJ with the Scala plugin). Scala can be quite sensitive regarding the used language version and things might break if you use the wrong one. For the assignment, we use *Scala 2.13.7* which is also specified in the build script. You can also run the tests included with the project skeleton using the build script. To run all tests execute the command `sbt test`.

Feature Model The configurable collection we implement follows the following feature model:



The following table explains each feature in more detail:

Table 1: Explanation for all features.

FIFO	Push elements to the front, pop elements from the bottom.
LIFO	Push to front, pop from front.
Priority	Always pop smallest element.
Capacity	Collection can only hold a certain number of elements.
Uniqueness	All elements in the collection are unique; Insertion only succeeds if no equivalent element is present.
Logging	Log all function calls.

[5] `https://www.scala-sbt.org/1.x/docs/`