23/11/2021

Scala in Practice

lab 05

Acceptance criteria:

Create Scala program with:

- Package *plugins* with abstractions for text manipulations:

```
trait Reverting = ??? //reverts text

trait LowerCasing = ??? //converts all chars in text to
lowerCase

trait SingleSpacing = ??? //removes all duplicated spaces in
text, example: “ala ma  kota” => “ala ma kota”

trait NoSpacing = ??? //remove all spaces in text, example:
“ala ma   kota” => “alamakota”

trait DuplicateRemoval = ??? //removes all chars which occur
more than once, example: “alzaa  cda” => “lzcd”

trait Rotating = ??? //rotates text once, example: “abc” =>
“cab”

trait Doubling = ??? //duplicates every second char in text,
example: “abcd” => “abbcdd”

trait Shortening = ??? //removes every second char in text,
example: “ab cd” => “a d”
```

- object *Actions* with abstractions for several actions using combinations of plugins *(ActionX.plugin()* should invoke the computation):

```
val actionA: Pluginable = ??? //plugin applying plugins with
order: SingleSpacing => Doubling => Shortening

val actionB: Pluginable = ???  //plugin applying plugins with
order: NoSpacing => Shortening => Doubling

val actionC: Pluginable = ??? //plugin applying plugins with
order: LowerCasing => Doubling

val actionD: Pluginable = ??? //plugin applying plugins with
order: DuplicateRemoval => Rotating
```

23/11/2021

Scala in Practice

lab 05

```
val actionE: Pluginable = ??? //plugin applying plugins with order: NoSpacing => Shortening => Doubling => Reverting

val actionF: Pluginable = ??? // plugin applying plugin Rotating 5-times
```

- `val actionG: Pluginable = ??? //plugin applying plugins with order: actionA => actionB`

- Create *application entry-point* object with some example tests for the above implementation

Note1: Assume that your code will be extensible in the future. Adding new plugins & actions should be strightforward
Note2: Dont use any **nulls** & **vars**

Michał Kowalczykiewicz