04 Lab
OOP in Scala

Mirko Viroli, Roberto Casadei
{mirko.viroli,roby.casadei}@unibo.it

C.D.L. Magistrale in Ingegneria e Scienze Informatiche
Alma Mater Studiorum—Università di Bologna, Cesena

a.a. 2020/2021

Lab 04: Outline

Outline

- Exercises on OOP in Scala

Getting started

- Fork/clone repository
  `https://github.com/unibo-pps/pps-20-21-lab04`
- Then, follow the instructions at the following slides

Exercise 1: complex numbers

Hints:

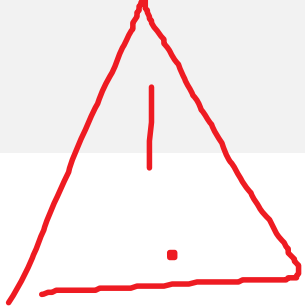
- Implement `Complex` with a `ComplexImpl` class, similar to `PersonImpl` in slides
- Implement `Complex.apply` (note that `???` is a valid Scala placeholder for an unimplemented method)
- Check that equality and `toString` do not work
- Use a `case class` `ComplexImpl` instead, creating objects without the `new` keyword
- Check equality and `toString` now

Exercise 2: students and courses

Hints:

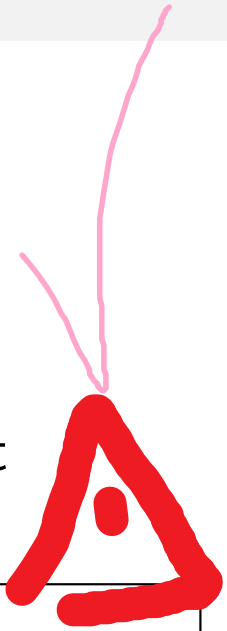
- Simply implement `Course`, e.g. with a `case class`
- Implement `Student` with a `StudentImpl` keeping a private `List` of courses
  - Use the list implementation used in Ex. 2
- Try to implement, in `StudentImpl`, method `courses` (with `map`)
- Try to implement, in `StudentImpl`, method `hasTeacher` (with `map` + `contains`)
  - Add the method `contains` to lists for the purpose
- Check your solution by running the given main program, comparing the actual output to the expected one
- Refactor the code so that method `enrolling` accepts a variable number of courses (variadic argument `Course*`)

Exercise 3: more oop



- Consider the following OOP exam:
  `https://bitbucket.org/mviroli/oop2018-esami/src/master/a05/e1/`
- and corresponding Java solutions:
  `https://bitbucket.org/mviroli/oop2018-esami/src/master/a05/sol1/`
- Goal: solve/implement it in idiomatic Scala code
- Take a look at the following files in the lab04 repository:
  `/src/u04lab/code/PowerIterators`
  `/test/u04lab/code/PowerIteratorsTest`

Optional exercises



- Implement a factory `List(e1,e2,...)` that builds a list `Cons(e1,Cons(e2,...))` out of the variable number of elements provided.
- Implement an extractor `sameTeacher(t)` on a list of `Courses` that extracts the teacher `t` in common to **all** courses (if any)

```
1 val courses = List(c1,c2,c3)
2 courses match {
3   case sameTeacher(t) => println(s"$courses have same
      teacher $t")
4   case _ => println(s"$courses have different teachers")
5 }
```

- NB: extractors (`unapply`) must use `scala.Option`, `scala.Some`... (it is easy to write a method to convert "our" `Option` to Scala's one)