Programming Paradigms
Lab 7

Jacek Wasilewski

Exercises

1. **Rational Number** – in the lecture, the idea of rational number data structure has been presented. Finish the implementation of the `Rational` class. Your class should have four methods: `add`, `sub`, `mul`, `div` for addition, subtraction, multiplication and division.
2. Create new file for your `Rational` class and move the code there.
3. Create a structure for a standalone application.
4. Use the standalone application to run examples of the `Rational` class. For this lab, use the standalone application to run any examples of use.
5. Define operators like `+`, `-`, `*` and `/` as shortcuts for `add`, `sub`, `mul` and `div`.
6. **Complex Number** – implement the data structure for complex numbers `https://en.wikipedia.org/wiki/Complex_number`. Requirements are:
   - creating complex number by defining its real part and imaginary part explicitly,
   - creating complex number by defining its real part only, imaginary part is 0,
   - basic arithmetic: addition, subtraction, multiplication, division,
   - all arithmetic operations should have shortcut operators: `+`, `-`, `*` and `/`,
   - string representation using the `toString` method; sample return: `5 + 6i`,

- real part and imaginary part should be doubles,
- square root – first implement method to get the modulus or the absolute value, it should be visible only to the `Complex` class,
- all input parameters should be checked appropriately.

7. **Number** – both `Complex` and `Rational` classes are numbers and they have similar structure. Create and abstract class (use name other than `Number` because it already exists) to store common members and/or operations.

8. Modify `Complex` and `Rational` classes to inherit from your number class – use `extends` keyword.

9. There is a trait called `Equals` that can be used to check if two objects are same or not. Use it in your abstract class and implement it in the `Complex` and `Rational`. Use `extends` keyword to use a trait.