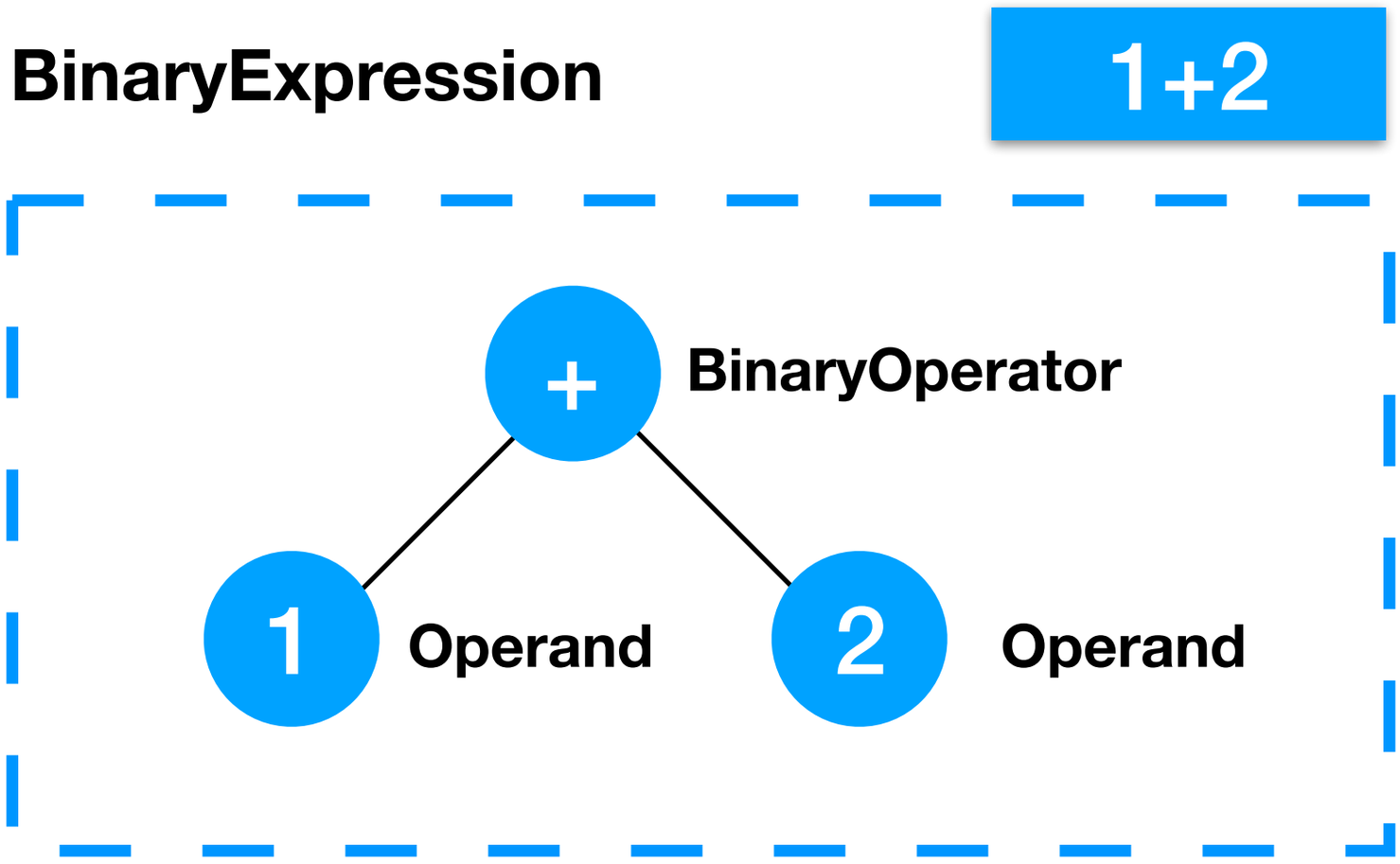
LAB5 - Expression (I)

LAB of COMP2021 OBJECT-ORIENTED PROGRAMMING

Objectives

- To build an expression library:
  - Create 3 Classes: Operand, BinaryOperator, BinaryExpression
- To create unit tests with JUnit

Simple Task



Simple Task: Requirements

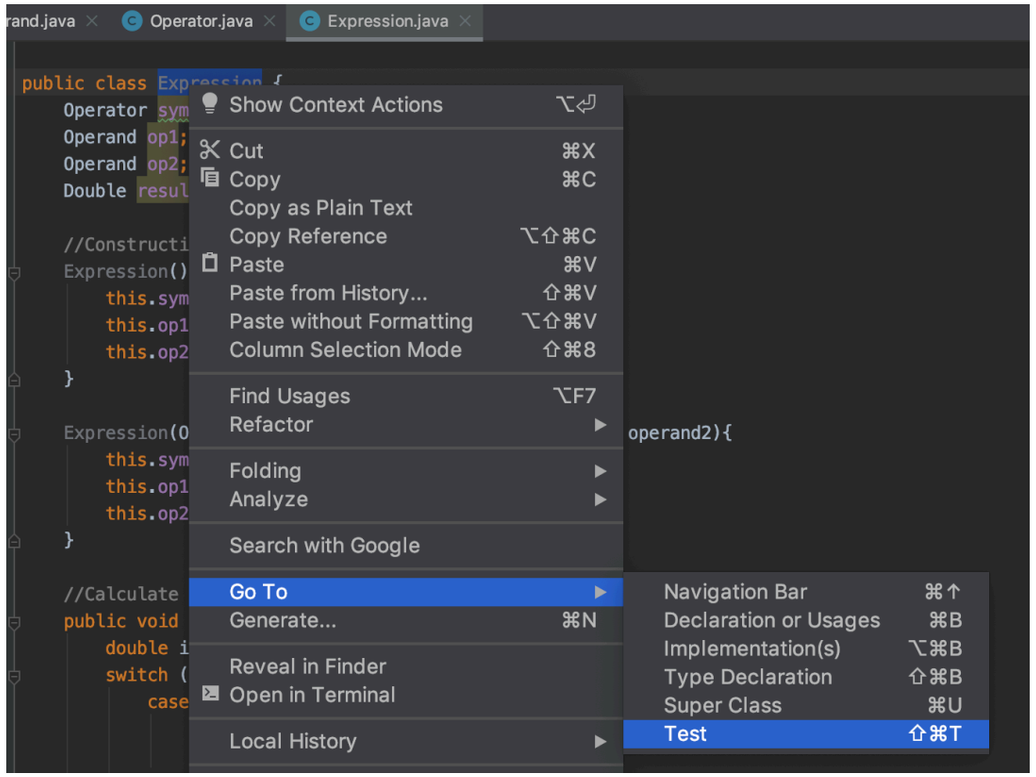
- Operand:
  - Parameterized Constructors
  - Method: int evaluate( )
- BinaryOperator:
  - Enum
  - Four Operations: +, -, *, /
  - Method: int calculate( )
- BinaryExpression:
  - Parameterized Constructors
  - Method: int evaluate( )
- You don’t have to include a main(String[] args) method in BinaryExpression Class.

Unit Testing with JUnit

- To test a software system is try to make it fail
- In testing, we run software system in a controlled way
- Unit Testing:
    - Testing of code units like functions, methods, classes, etc.

Unit Testing with JUnit

- JUnit is a simple framework to write unit tests
- Simple way to use JUnit in IntelliJ IDEA:

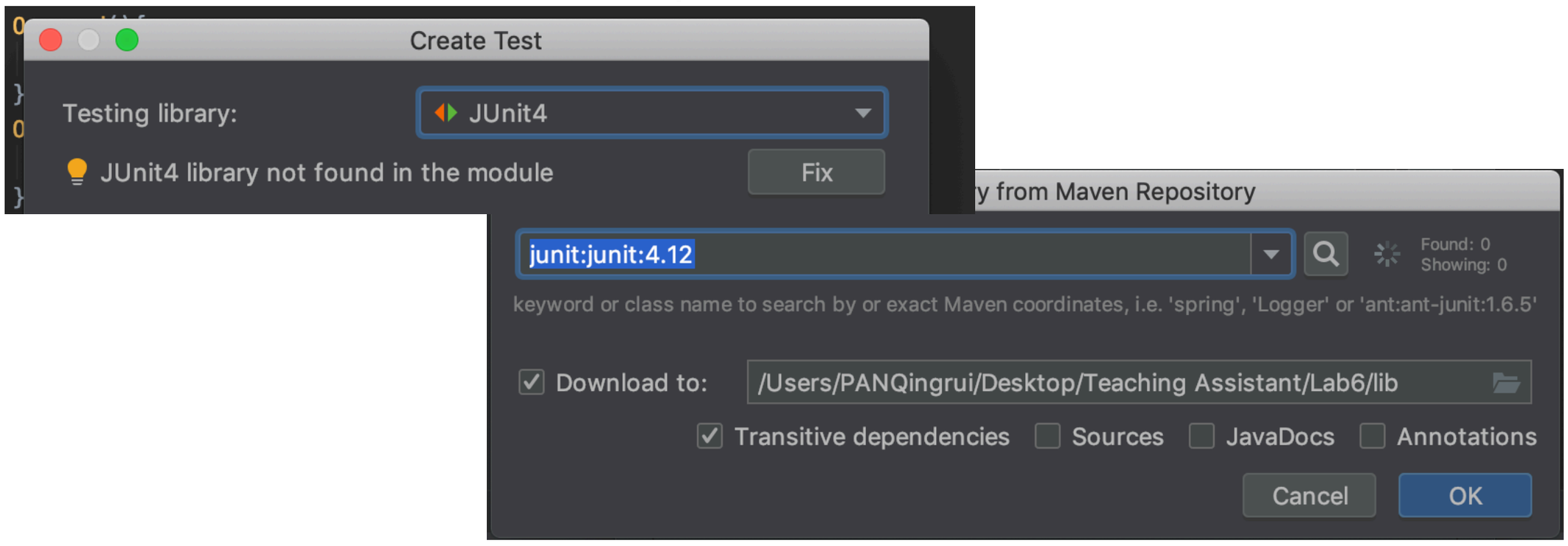


Unit Testing with JUnit

- @Test (@Before, @After ,etc.):
  - IntelliJ IDEA import for you automatically.

```
import org.junit.jupiter.api.Test;
```

- Import the JUnit Lib with the help of IntelliJ IDEA



Unit Testing with JUnit

- Try some test methods!

    1. create a Test method;
    2. instantiate objects;
    3. invoke methods on the objects;
    4. compare method results with your expected ones using Assert* methods

Unit Testing with JUnit

- Assert
  - Assert is a method useful in determining Pass or Fail status of a test case.
- Useful assert methods:
  - assertTrue(condition)
  - assertNull(object)
  - assertEquals(expected, actual)
  - ……

- Thank you

Additional Task

- Test classes Rational and Complex in Assignment 2!
- Examples: Test the Add( ) in Rational class

1. Create a test class
2. Create a test method
3. Assert* the result