

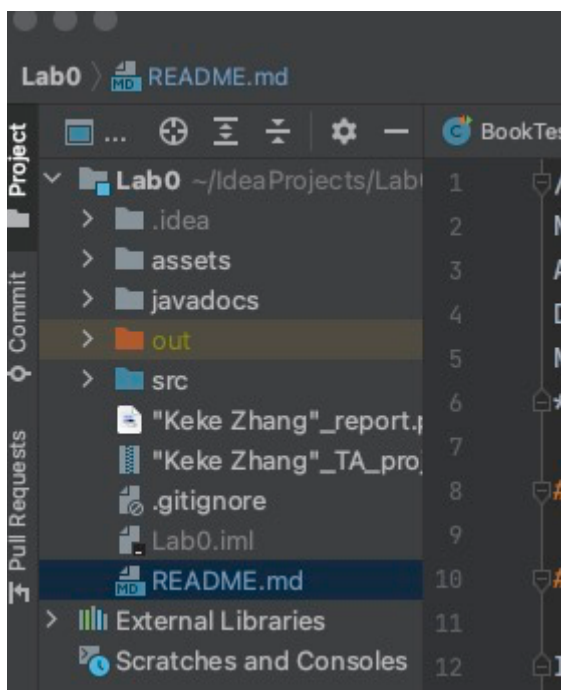
Report

1 reflection

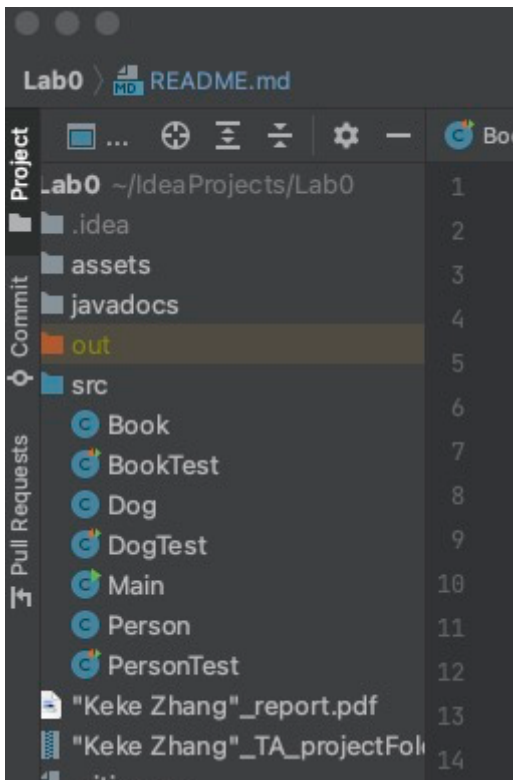
I get familiar with the three ways to compile a program, learn the basics of running a program and how to create class object and run it.

2 Screenshots

create a project in my IDE



add Book.java and Person.java to my project



add the PersonTest.java to my project and verify that all the PersonTests test cases pass.

The screenshot shows an IDE with a project named 'Lab0'. The left sidebar displays the project structure, including a 'src' directory with files like 'Book', 'BookTest', 'Dog', 'DogTest', 'Main', 'Person', and 'PersonTest'. The 'PersonTest' file is selected and open in the editor. The code is a JUnit test class for the 'Person' class, featuring imports for 'org.junit.Before', 'org.junit.Test', and 'org.junit.Assert.assertEquals'. It includes a class-level comment and a default constructor. The bottom panel shows the 'Run' configuration for 'PersonTest' and the execution results: 'Tests passed: 3 of 3 tests - 2 ms'. The test methods 'testFirst', 'testYearOfBirth', and 'testSecond' are listed with their respective execution times (1ms, 1ms, and 0ms).

```
1  /*****
2      Name: Keke Zhang
3      Assignment: Lab 0
4      Date: 14/Jan/2023
5      Notes: PersonTest
6  *****/
7
8  import org.junit.Before;
9  import org.junit.Test;
10
11  import static org.junit.Assert.assertEquals;
12
13
14  /**
15   * A JUnit test class for the Person class.
16   */
17  public class PersonTest {
18      /**
19       * default constructor
20       */
21      public PersonTest(){
22
23      }
24  }
```

Run: PersonTest x

Tests passed: 3 of 3 tests - 2 ms

Test Method	Execution Time
testFirst	1 ms
testYearOfBirth	1 ms
testSecond	0 ms

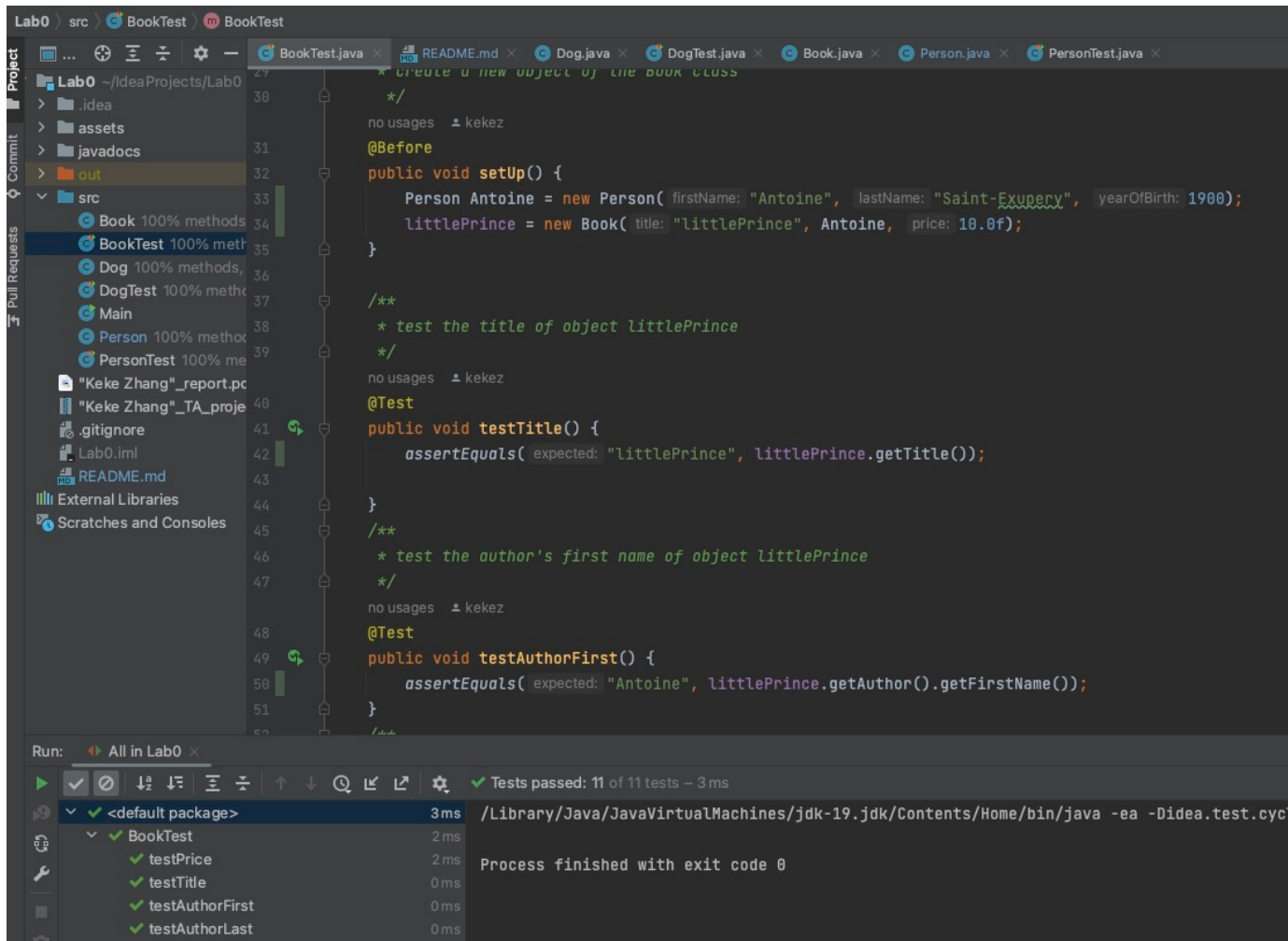
Process finished with exit code 0

view the level of coverage PersonTests currently offers.

The screenshot shows the 'Coverage' tool window in the IDE, displaying the coverage for the 'PersonTest' class. The table lists the coverage for the 'Person' class and the 'PersonTest' class, showing 100% coverage for all elements.

Element	Class, %	Method, %	Line, %
all			
Person	100% (1/1)	100% (4/4)	100% (7/7)
PersonTest	100% (1/1)	100% (5/5)	100% (5/5)

create a JUnit test class for the Book class and all the public methods of the Book class work as expected.



checking the level of coverage

Element	Class, %	Method, %	Line, %
all			
Book	100% (1/1)	100% (4/4)	100% (7/7)
BookTest	100% (1/1)	100% (6/6)	100% (7/7)
Dog	100% (1/1)	100% (5/5)	100% (9/9)
DogTest	100% (1/1)	100% (6/6)	100% (6/6)
Person	100% (1/1)	100% (4/4)	100% (7/7)
PersonTest	100% (1/1)	100% (5/5)	100% (5/5)

3 Extension

I create my own class object and test class (Dog.java and DogTest.java) I submit my con code as a Github link instead.

4 Grading Statement

Based on the rubric, i think i could get 28-29