LATVIJAS UNIVERSITĀTE
ANNO 1919

NACIONĀLAIS ATTĪSTĪBAS PLĀNS 2020

EIROPAS SAVIENĪBA
Eiropas Sociālais fonds

IEGULDĪJUMS TAVĀ NĀKOTNĒ

Java programmēšanas pamati

VADĪBAS UN UZŅĒMĒJDARBĪBAS MĀCĪBU CENTRS

11. NODARBĪBA

KODA TESTĒŠANA

KODA TESTĒŠANA

PĀRSKATS

TESTĒŠANAS JĒGA

- Tests ir programma, kas izpilda citu programmu, lai pārliecinātos, ka kods pareizs
- Tests var pārbaudīt:
  - Sagaidāmo stāvokli (stāvokļa testēšana)
  - Sagaidāmo notikumu secību (uzvedības testēšana)
- Augsts testa pārklājums ļauj izstrādāt jaunu funkcionalitāti, nebaidoties sabojāt esošo

PRIMĀRĀS PROGRAMMATŪRAS PĀRBAUDES JOMAS

- Vienībtesti (unit tests)
  - Paredzēti nelielai koda daļai (metode, klase)
  - Ārējās klases atkarības tiek aizstātas ar to “mok” objektiem (mocks)
- Integrācijas testi (integration tests)
  - Mērķis pārbaudīt komponentes darbību vai integrāciju starp vairākām komponentēm
  - Pārbauda vai visa sistēma darbojas kā paredzēts

(1) MANUĀLĀ TESTĒŠANA

- Koda tetēšana manuāli, neizmantojot speciālu programmatūru
- Laikietilpīgi un garlaicīgi
  - Tā kā testēšanu veic cilvēks, tad tas notiek ļoti lēni un garlaicīgi
- Milzīgas investīcijas cilvēkresursos
  - Tā kā testēšana ir jāveic manuāli, tad tai ir nepieciešams vairāk testētāju

(2) MANUĀLĀ TESTĒŠANA

- Nav uzticama
  - Manuālā testēšana ir mazāk uzticama, jo tajā jāņem vērā cilvēku kļūdas
- Neprogrammējama:
  - Nav iespējams uzrakstīt testu, kurš pārbaudītu slēptu informāciju

AUTOMĀTISKIE TESTI

- Testu izpilde izmantojot testu automatizācijas rīku, tiek saukta par testu automatizāciju
- Automatizēta testu izpildīšana norit ievērojami ātrāk kā cilvēks to var paveikt
- Mazākas investīcijas cilvēkresursos
  - Cilvēkresursi ir nepieciešami tikai testu uzrakstīšanai, nevis veikšanai
- Automātiskie testi ir precīzi un uzticami
- Programmējami
  - Testētāji var uzrakstīt testus, kas pārbauda lietotājam slēptus datus

JUNIT

SISTĒMA

VIENĪBTESTĒŠANAS PIEEJA

- **JUnit** ir vienībtestēšanas sistēma Java programmēšanas valodai
- **JUnit tests** ir klases (saukta arī par testa klasi) metode, kas paredzēta tikai testēšanai
- Vienībtestu raksturo:
  - Zināmi ievaddati
  - Sagaidāmais rezultāts

PIEMĒRS: TESTĒJAMĀ KLASE

```
public class Calculator {
        public int sum(int a, int b) {
                return a + b;
        }
}
```

PIEMĒRS: TESTA KLASE

```
public class CalculatorTest {

        private Calculator victim;

        @Before
        public void setUp() {
                victim = new Calculator();
        }

        @Test
        public void shouldCalculateSum() {
                int result = victim.sum(3, 5);
                assertEquals(8, result);
        }
}
```

POPULĀRĀKĀS JUNIT ANOTĀCIJAS: TESTA DEKLARĒŠANAI

Anotācija	Apraksts
`@Test` `public void testCase() {}`	Anotācija @Test norāda, ka sekojošā metode ir testa metode
`@Test(expected = Exception.class)` `public void testCase() {}`	Ja metode nemet norādīto izņēmumu tad tests ir izgāzies (fail)
`@Test(timeout = 500)` `public void testCase() {}`	Ja metode izpildās ilgāk par 500 milisekundēm, tad tests ir izgāzies (fail)
`@Ignore` `public void testCase() {}`	Anotācija ir noderīga, ja vēlaties īslaicīgi atslēgt, kāds specifiska testa izpildi

POPULĀRĀKĀS JUNIT ANOTĀCIJAS: TESTA SAGATAVOŠANAI

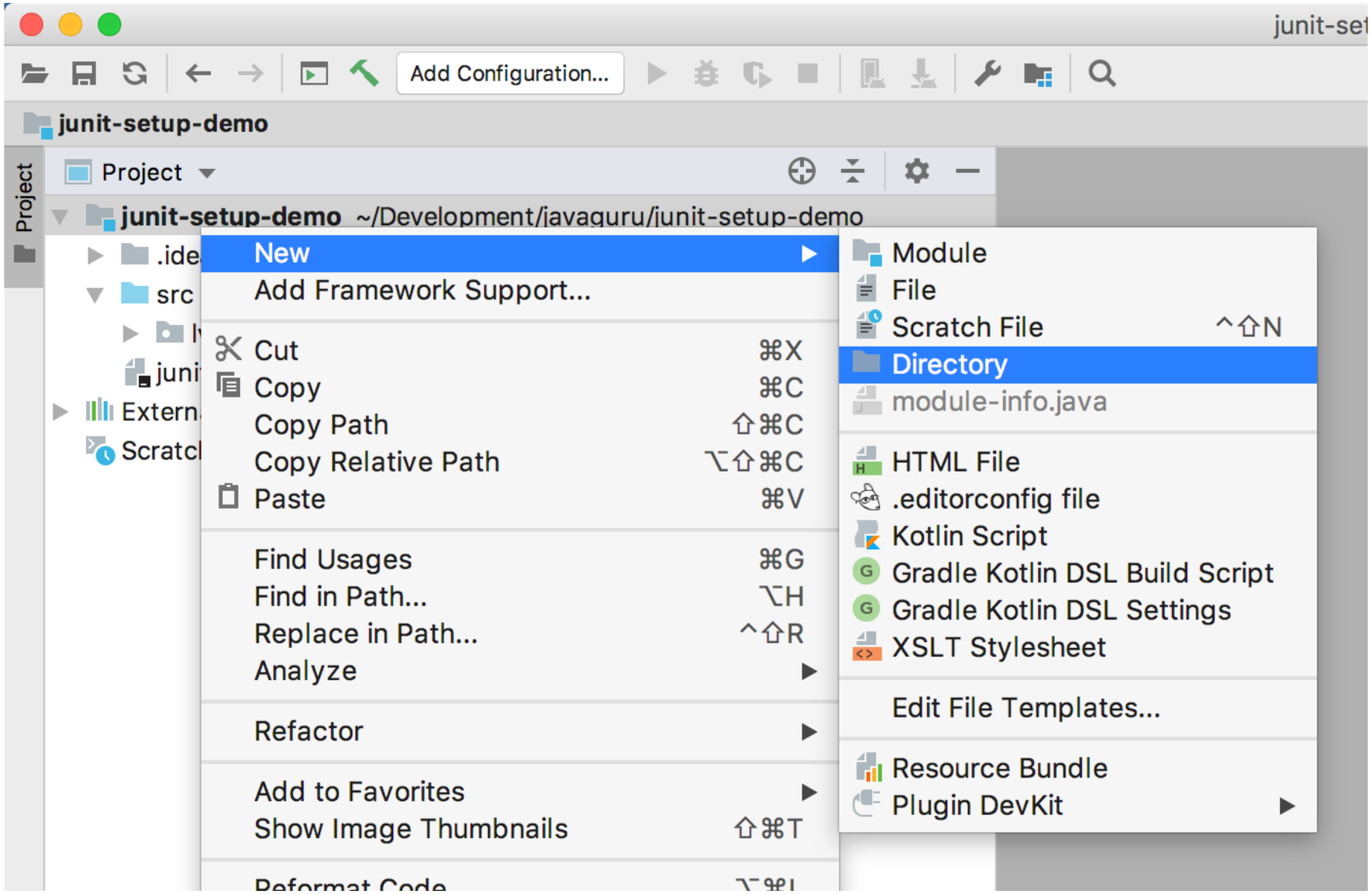
Anotācija	Apraksts
`@Before` `public void setUp() {}`	Šī metode tiek izpildīta pirms katra testa
`@After` `public void tearDown() {}`	Šī metode tiek izpildīta pēc katra testa
`@BeforeClass` `public void setUp() {}`	Šī metode tiek izpildīta vienu reizi pirms visu testu izpildes
`@AfterClass` `public void tearDown() {}`	Šī metode tiek izpildīta vienu reizi pēc visu testu izpildes

POPULĀRĀKĀS ASSERT IZTEIKSMES

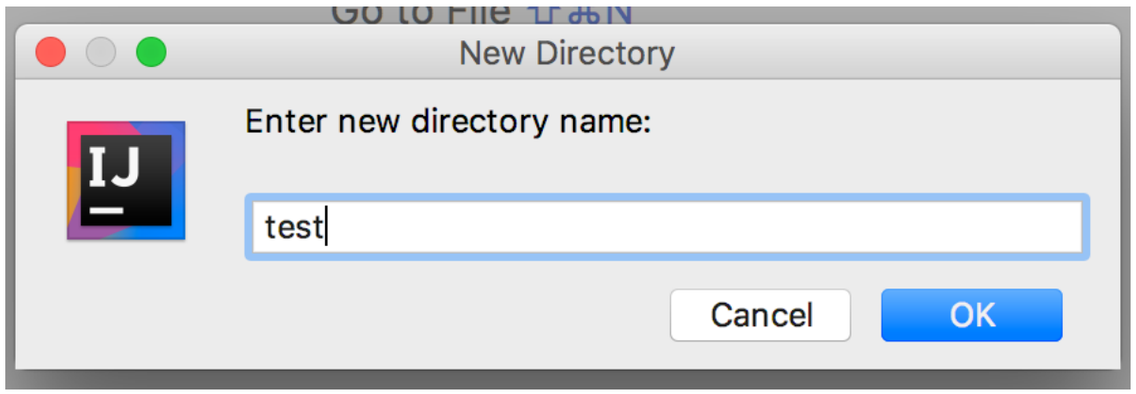
Anotācija	Apraksts
Assert.assertEquals(expected, actual); Assert.assertNotEquals(expected, actual);	Pārbauda vai dotie objekti ir vienādi, pēc equals principa
Assert.assertTrue(actual); Assert.assertFalse(actual);	Pārbauda būla tipa resultātu, attiecīgi paties vai nepatiess
Assert.assertNull(actual); Assert.assertNotNull(actual);	Pārbauda vai resultāts ir tukšs, vai netukšs – nav null
Assert.assertSame(expected, actual); Assert.assertNotSame(expected, actual);	Pārbauda vai abas objektu norādes, norāda uz vienu un to pašu objektu atmiņā

MANUALA PROJEKTA IESTATĪŠANA

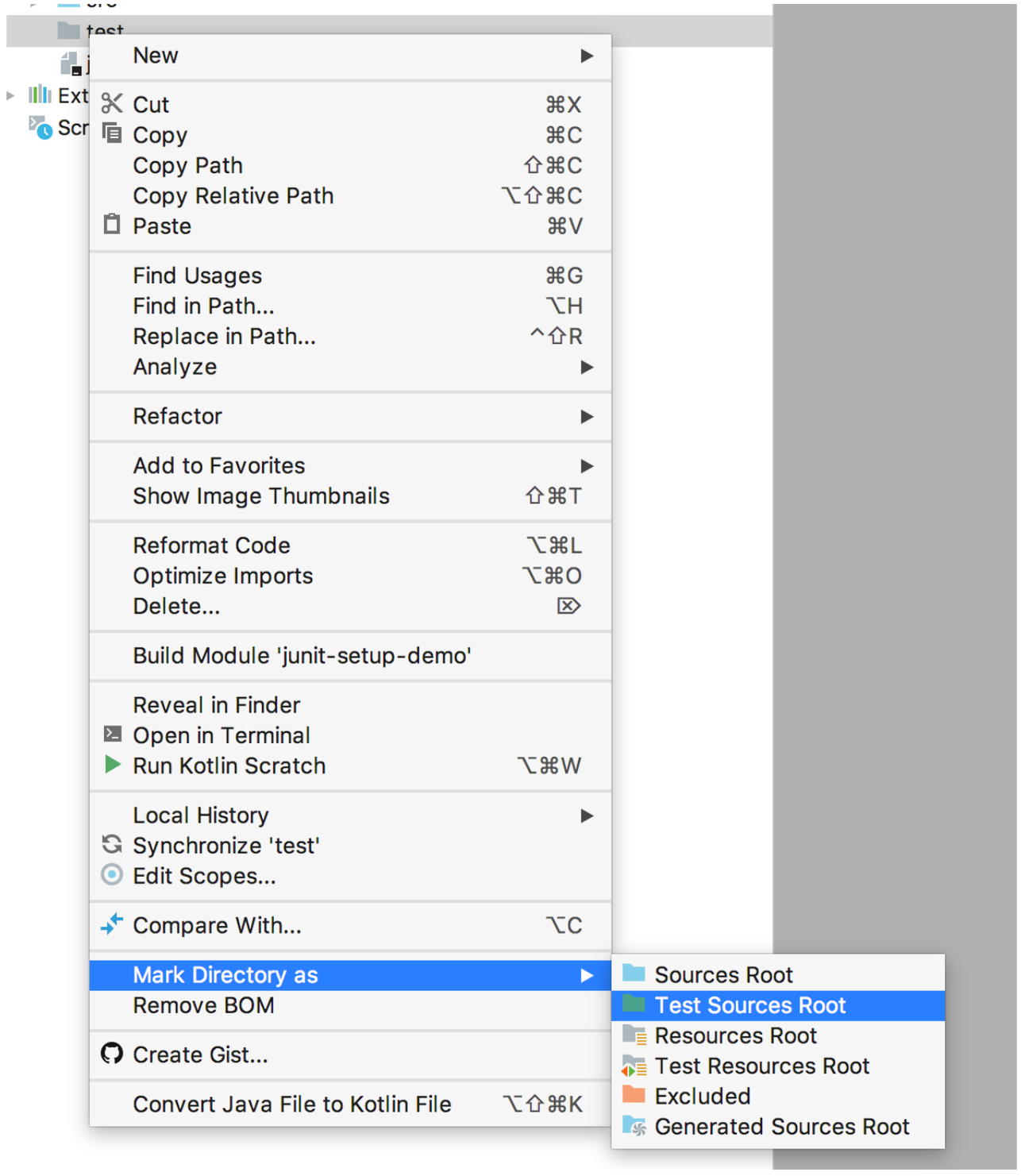
1: IZVEIDO JAUNU PROJEKTU



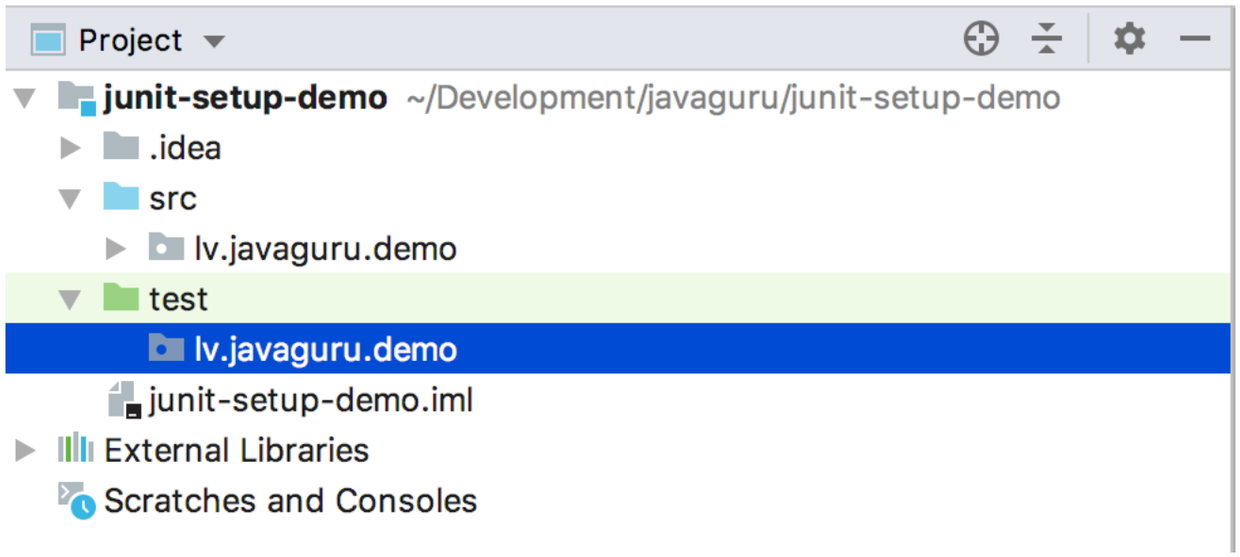
2: DIREKTORIJU NOSAUC “TEST”



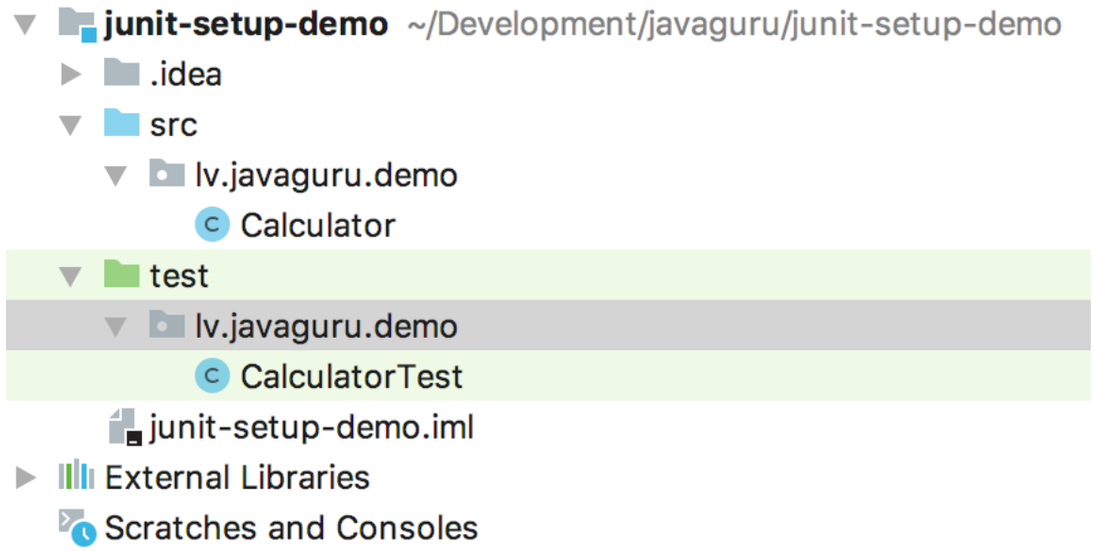
3: ATZĪMĒ TO KĀ TESTU DIREKTORIJU



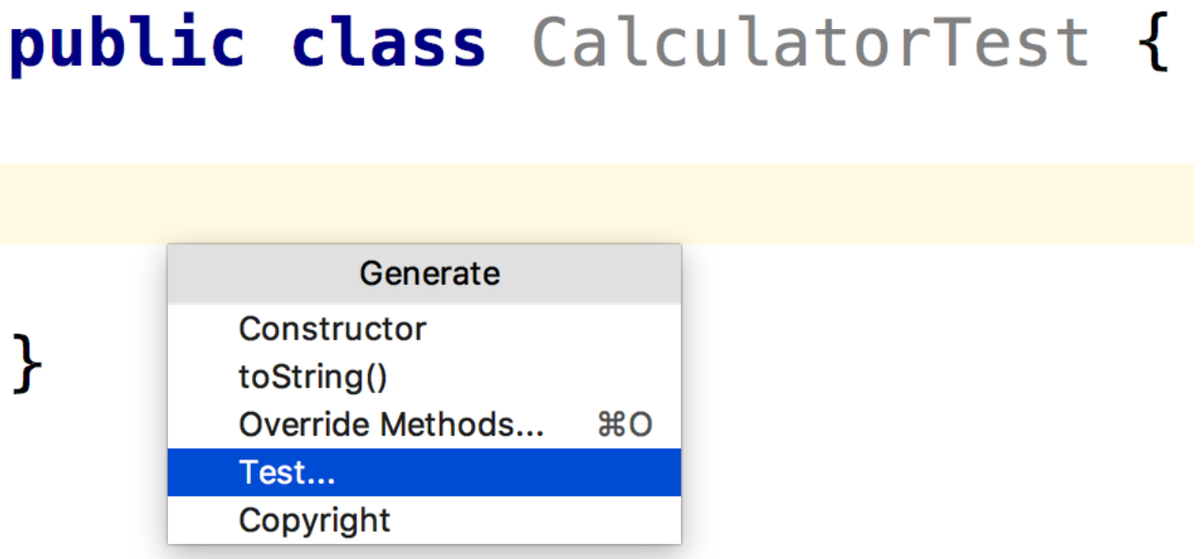
4: IZVEIDO DUBLĒJOŠU PAKOTNI



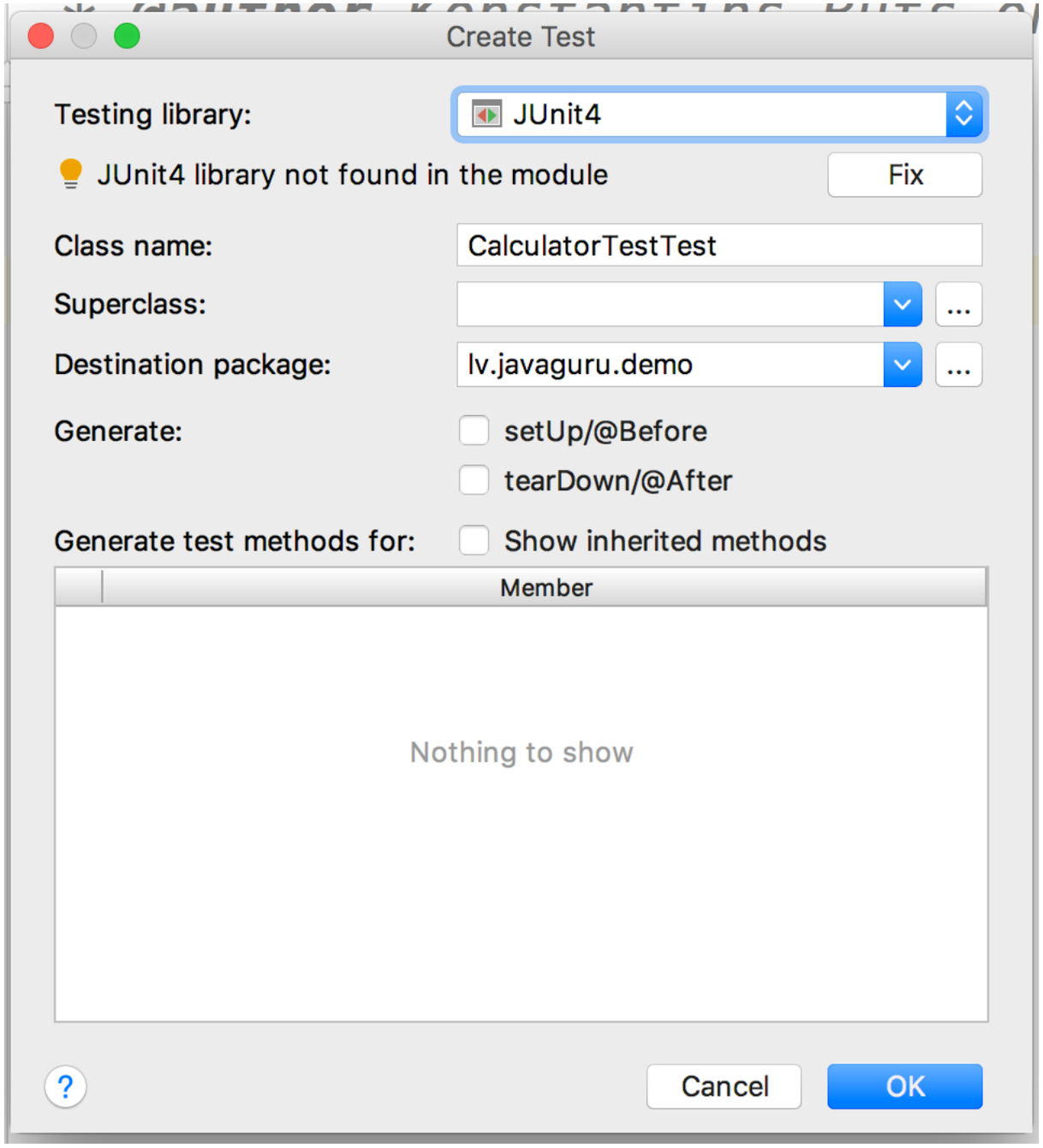
5: IZVEIDO TESTA KLASI



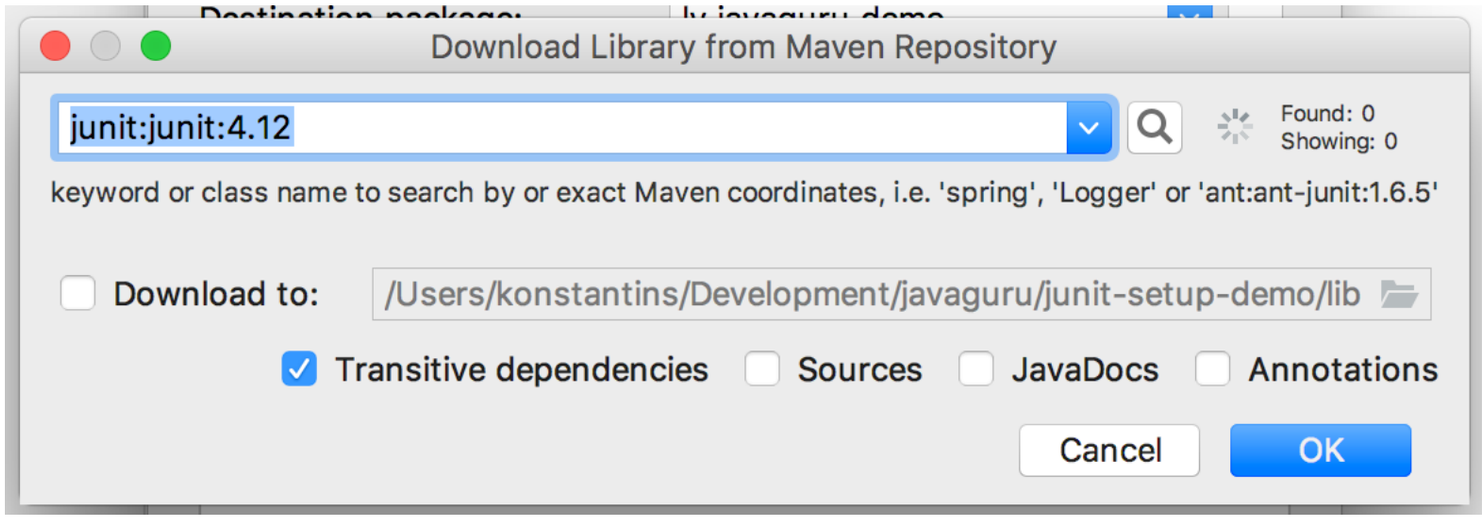
6: ĢENERĒ TESTU TESTA KLASĒ



7: PIEVIENO TESTĒŠANAS BIBLIOTĒKU



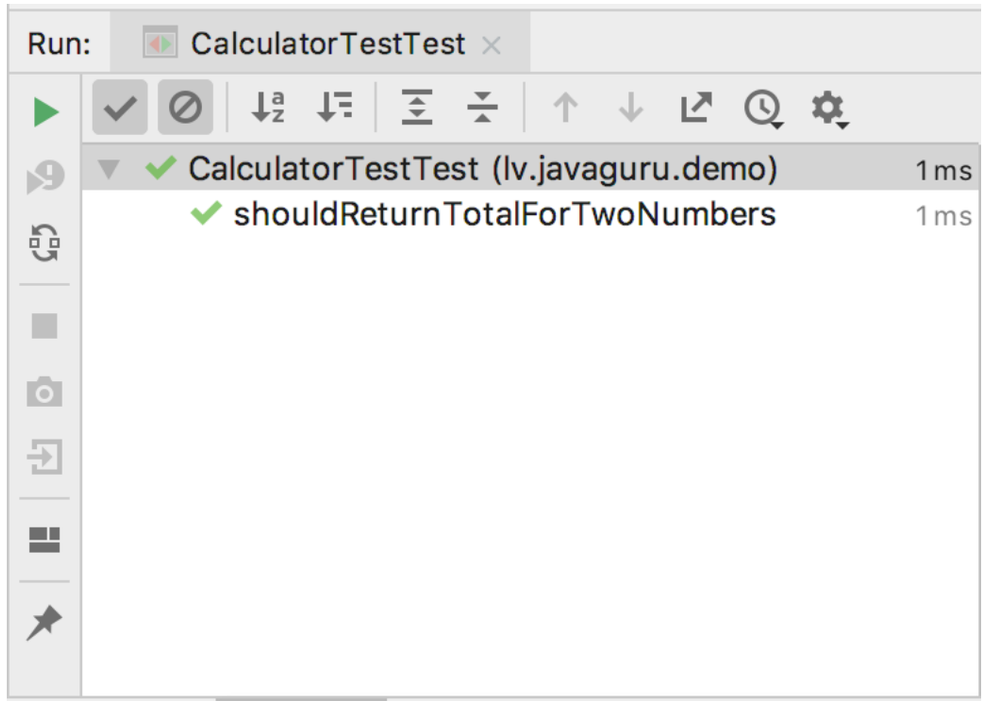
8: LEJUPLĀDĒ BIBLIOTĒKU



9: UZRAKSTI TESTA SCENĀRIJU

```
12  public class CalculatorTestTest {
13
14      private Calculator victim;
15
16      @Before
17      public void setUp() {
18          victim = new Calculator();
19      }
20
21      @Test
22      public void shouldReturnTotalForTwoNumbers() {
23          int result = victim.sum( ...values: 2, 3);
24          Assert.assertEquals( expected: 5, result);
25      }
26
27  }
28
```

10: PALAID TESTUS



ATSAUCES

- https://junit.org/junit4/
- https://www.tutorialspoint.com/junit/junit_basic_usage.html
- http://www.vogella.com/tutorials/JUnit/article.html
- https://www.swtestacademy.com/junit-tutorial/
- https://dzone.com/articles/7-popular-unit-test-naming

LATVIJAS UNIVERSITĀTE
ANNO 1919

VADĪBAS UN UZŅĒMĒJDARBĪBAS MĀCĪBU CENTRS