

## Laboration: JUnit och testning

Testdriven utveckling i Java

Uppdaterad 2020-10-28

### Syfte

Laboration, där studenten ska följa instruktioner för att sätta upp JUnit 5 ramverken i sina miljöer och sen skapa testscenarion passande JUnits metodik.

### *Denna laboration syftar till att uppfylla följande:*

Den studerande ska få en grundläggande insyn och kunskap om hur man sätter upp en testmiljö i utvecklingsverktyget Eclipse och exekvera enklare enhetstester.

### Inlämning

Inlämning för denna laboration sker på Google Classroom. Studenten ska via Github lämna in deras kod där de använder ramverket JUnit 5 för att lösa uppgifter.

### Betygskriterie

Studenten anses att ha avklarat, och därav är godkänd, laborationen vid inlämningen av deras kod som använder ramverket JUnit 5, inom tidsramen.

Betyg: IG/G

### Tidsram

Laborationen varar tills 15:00 den 2020-10-05.

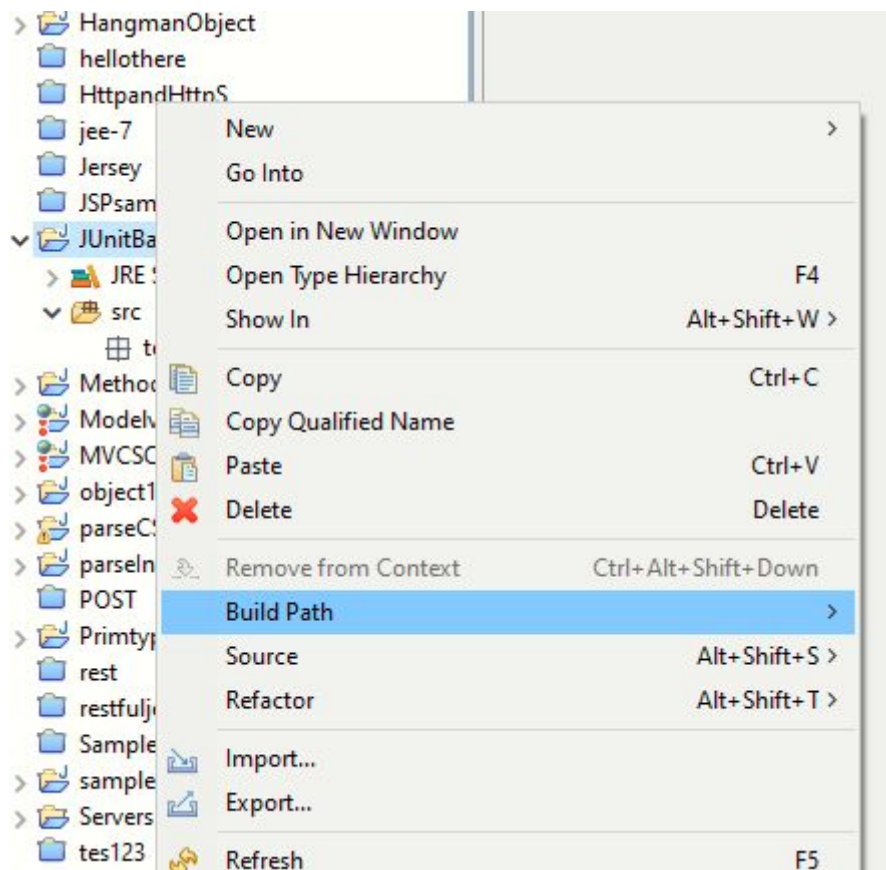
### Hjälpmedel

Studenten ska främst ta sig hjälp av officiella källor för JUnit 5 :

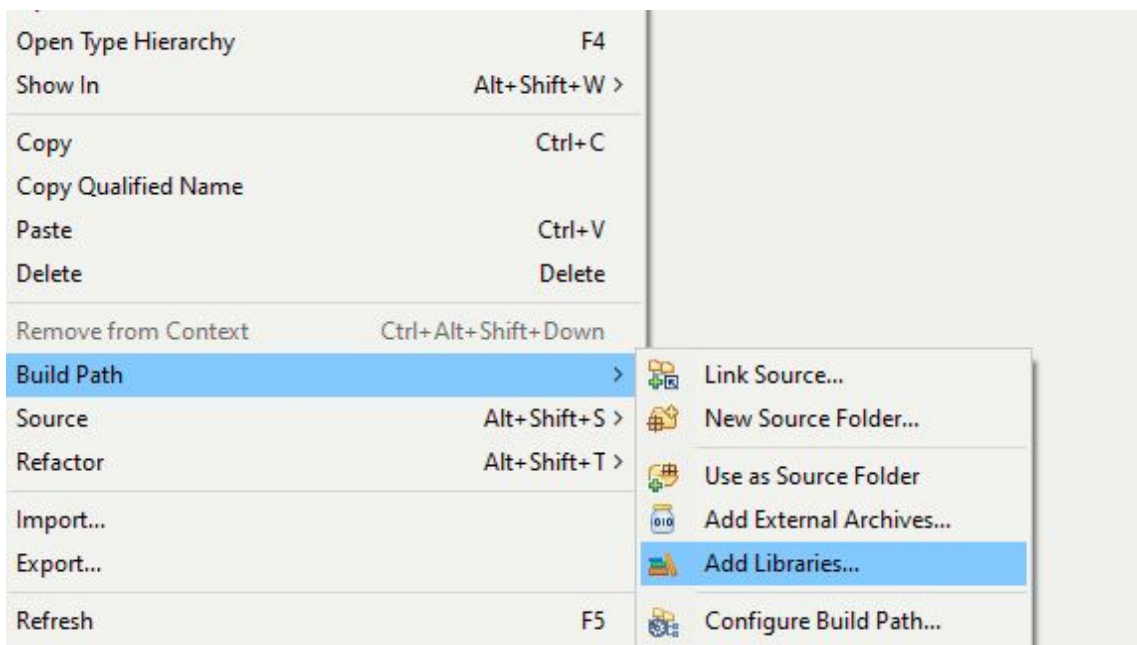
<https://junit.org/junit5/docs/current/user-guide/>

Hjälp från andra studenter, andra källor och av lärare är tillåtet, men först efter studenten kollat officiella källor.

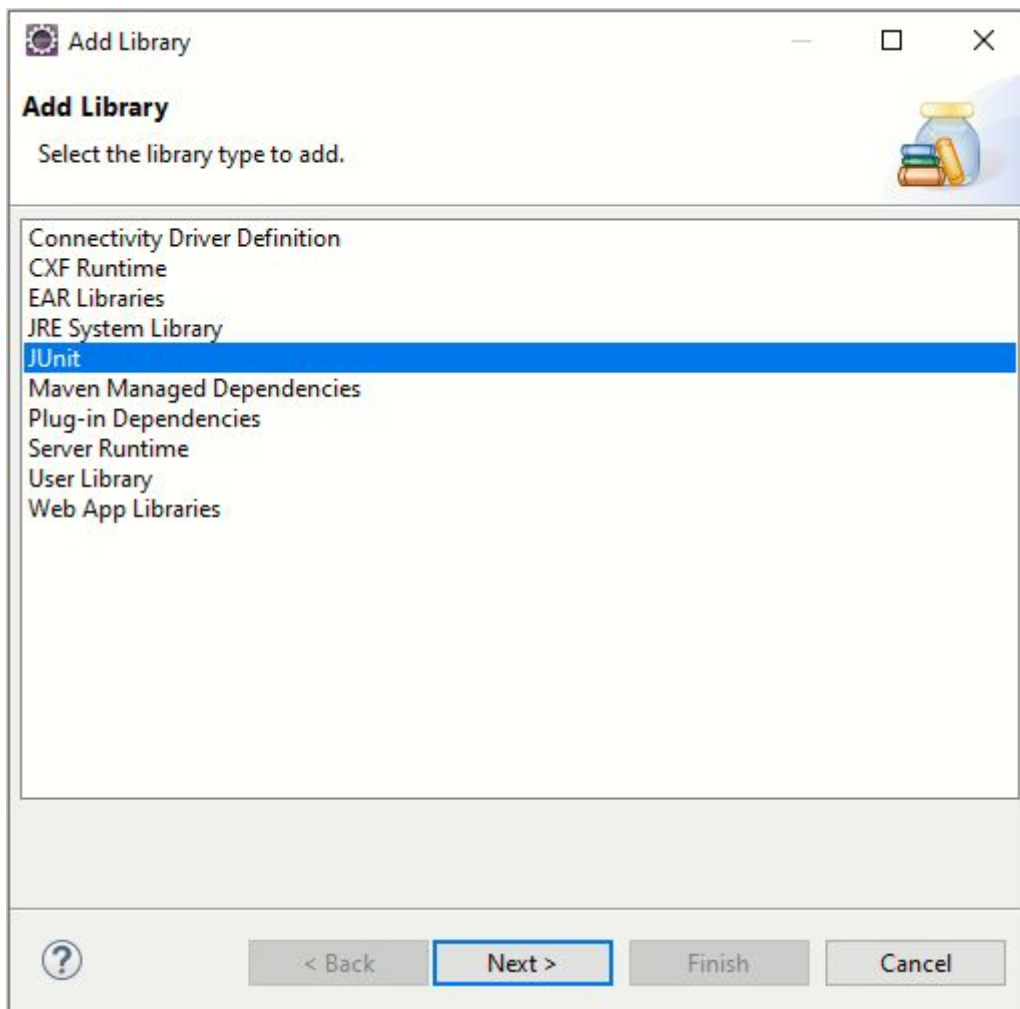
## Laboration Steg 1: Lägg till JUnit 5 i ett projekt i Eclipse



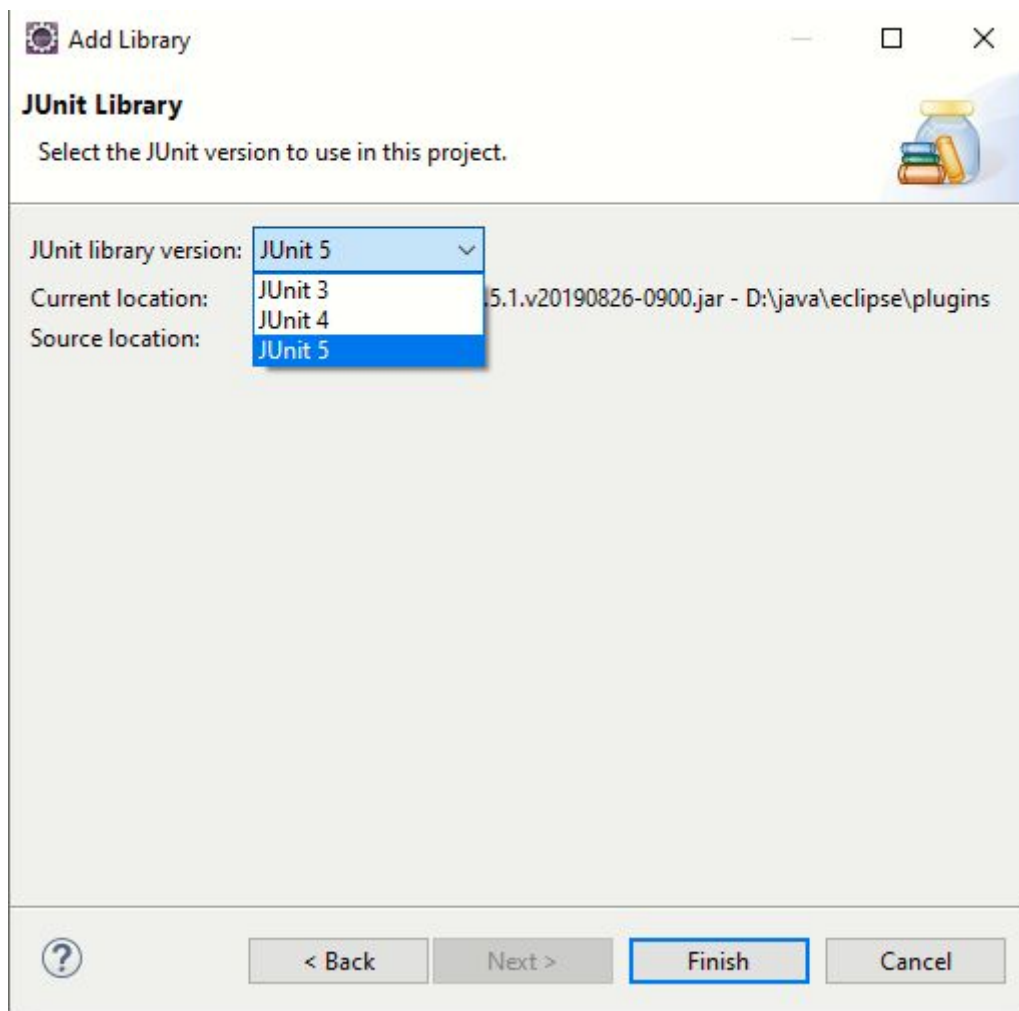
- Välj ett projekt eller starta ett nytt
- Högerklicka på projektet för att få upp en meny
- Välj Build Path



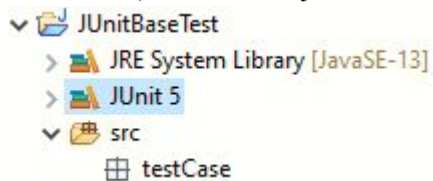
- Under Build Path välj Add Libraries



- Välj JUnit



- Välj versionen av JUnit som ska köras (5 i detta fall)



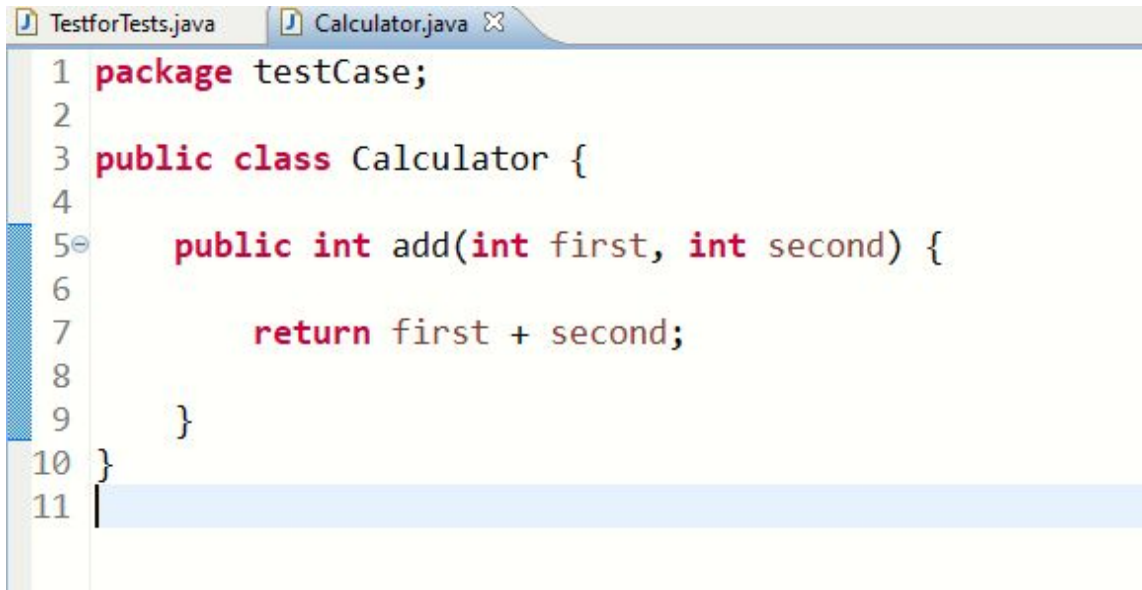
- Se till att JUnit 5 är i ert projekt

```
TestfforTests.java x
1 package testCase;
2
3 import org.junit.Ignore;
4 import org.junit.jupiter.api.Test;
5 import org.junit.jupiter.migrationsupport.EnableJUnit4MigrationSupport;
6
7 // @ExtendWith(IgnoreCondition.class)
8 @EnableJUnit4MigrationSupport
9 class IgnoredTestsDemo {
10
11     @Ignore
12     @Test
13     void testWillBeIgnored() {
14     }
15
16     @Test
17     void testWillBeExecuted() {
18     }
19 }
```

- Skapa en Java Klass med kod från JUnit för att testa att det ligger med i ert projekt
- Om inget är rött, så är JUnit i ert projekt

## Laboration Steg 2: Första testet

- Skapa först något att testa



```
1 package testCase;
2
3 public class Calculator {
4
5     public int add(int first, int second) {
6
7         return first + second;
8
9     }
10 }
11
```

- Skapa en Klass och döp den till Calculator
- Skapa en metod med namnet add i klassen
- Gör så att den tar emot 2 ints och returnerar summan av dem

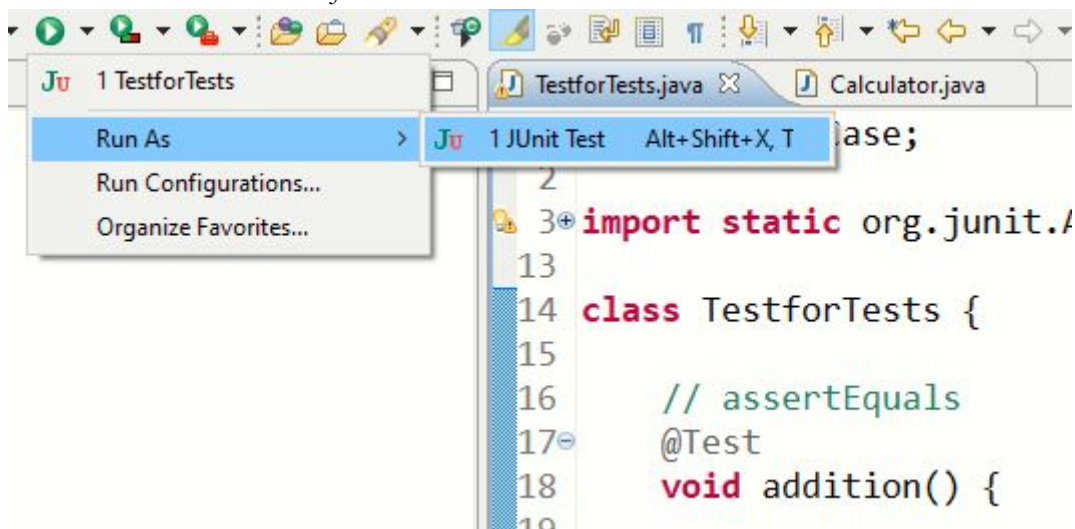
- Skapa ännu en Klass att göra era tester i

```

1 package testCase;
2
3 import static org.junit.Assert.assertEquals;
4
13
14 class TestforTests {
15
16     // assertEquals
17     @Test
18     void addition() {
19
20         System.out.println(" adding test");
21         Calculator calculator = new Calculator();
22         assertEquals(2, calculator.add(1, 1));
23     }
24
25
26
27
28
29 }

```

- Där skapa en metod
- Lägg "@Test" just över metoden
- gör ett "assertEquals" test mot er metod i Calculator
- kör er kod som ett "JUnit test"



```

Jv 1 TestforTests
Run As > Jv 1 JUnit Test Alt+Shift+X, T
Run Configurations...
Organize Favorites...

```

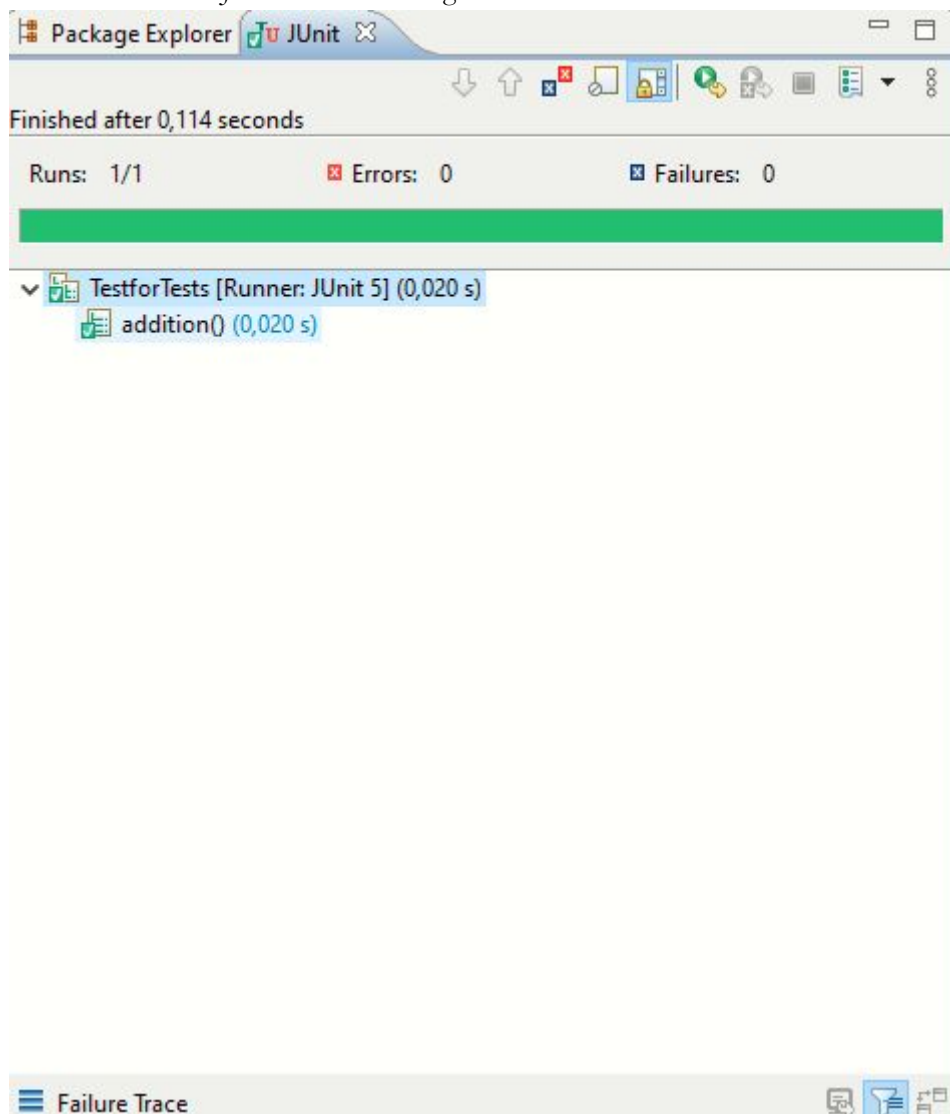
```

1 package testCase;
2
3 import static org.junit.Assert.assertEquals;
4
13
14 class TestforTests {
15
16     // assertEquals
17     @Test
18     void addition() {
19
20         System.out.println(" adding test");
21         Calculator calculator = new Calculator();
22         assertEquals(2, calculator.add(1, 1));
23     }
24
25
26
27
28
29 }

```



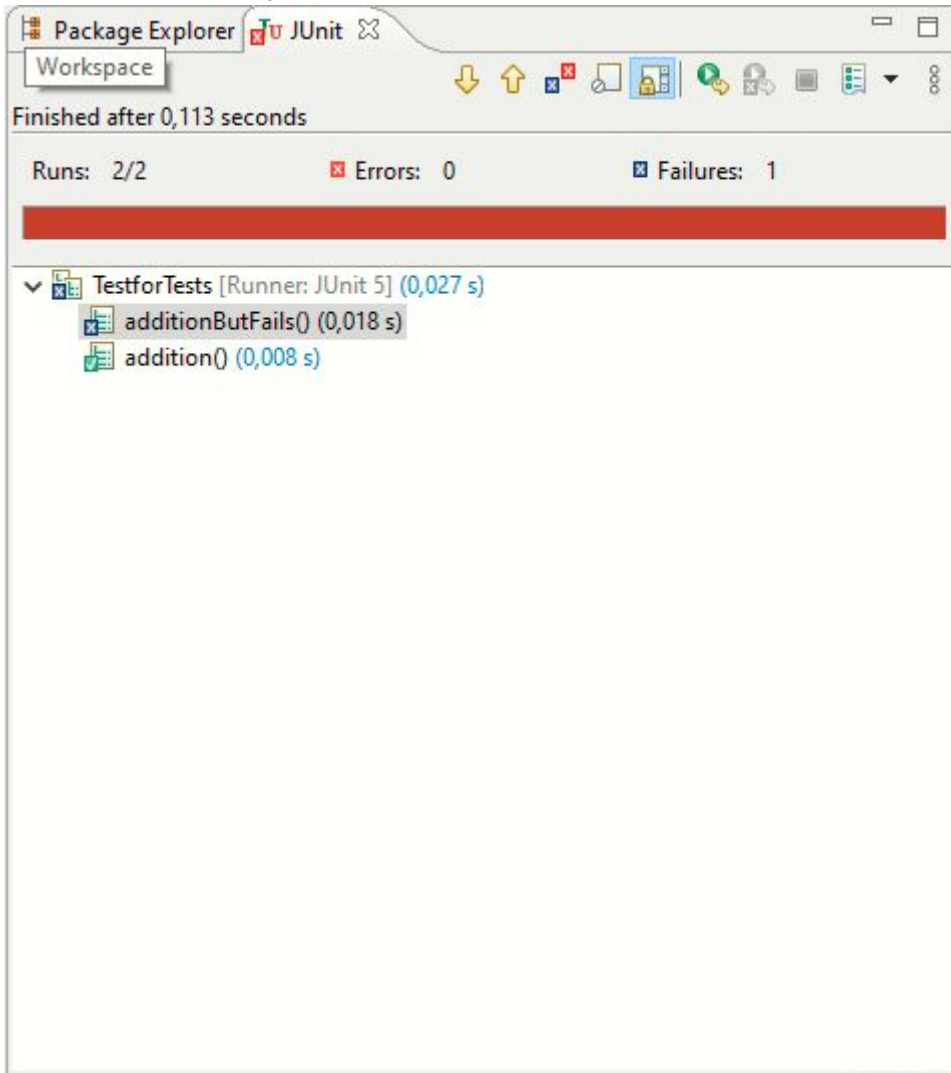
- Se ifall ert JUnit test klarade sig



- Skapa ett nytt test för samma kod, men nu gör så att koden inte klarar testet.

```
1 package testCase;
2
3 import static org.junit.Assert.assertEquals;
13
14 class TestforTests {
15
16     // assertEquals
17     @Test
18     void addition() {
19
20         System.out.println(" adding test");
21         Calculator calculator = new Calculator();
22         assertEquals(2, calculator.add(1, 1));
23     }
24     @Test
25     void additionButFails() {
26
27         System.out.println(" adding failing test");
28         Calculator calculator = new Calculator();
29         assertEquals(2, calculator.add(2, 1));
30     }
31 }
```

- Kolla resultat i JUnit rutan



- Grattis! Du har just gjort ditt första test.

### Laboration Steg 3: Officiell dokumentation

- Kolla upp vad "Annotation" och vad "@Test" är till för
- Sen skapa vad JUnit kallar "standard test class"
- Kolla upp vad "Assertion" är och vad assertEquals gör
- Kolla upp olika typer av assertEquals metoder med olika parametrar (metod signaturer)
- Skapa en enhetstest som testar om två long datatyper är samma och som samtidigt skriver ut ett meddelande när testet körs

### Laboration Steg 4: Era egna tester

- Skapa ett test som använder någon annan "Assertion" metod än assertEquals
- Fundera ut vad som skiljer sig åt mellan assertEquals och assertTrue
- Skriv en testmetod med hjälp av assertTrue som undersöker om två String datatyper är samma