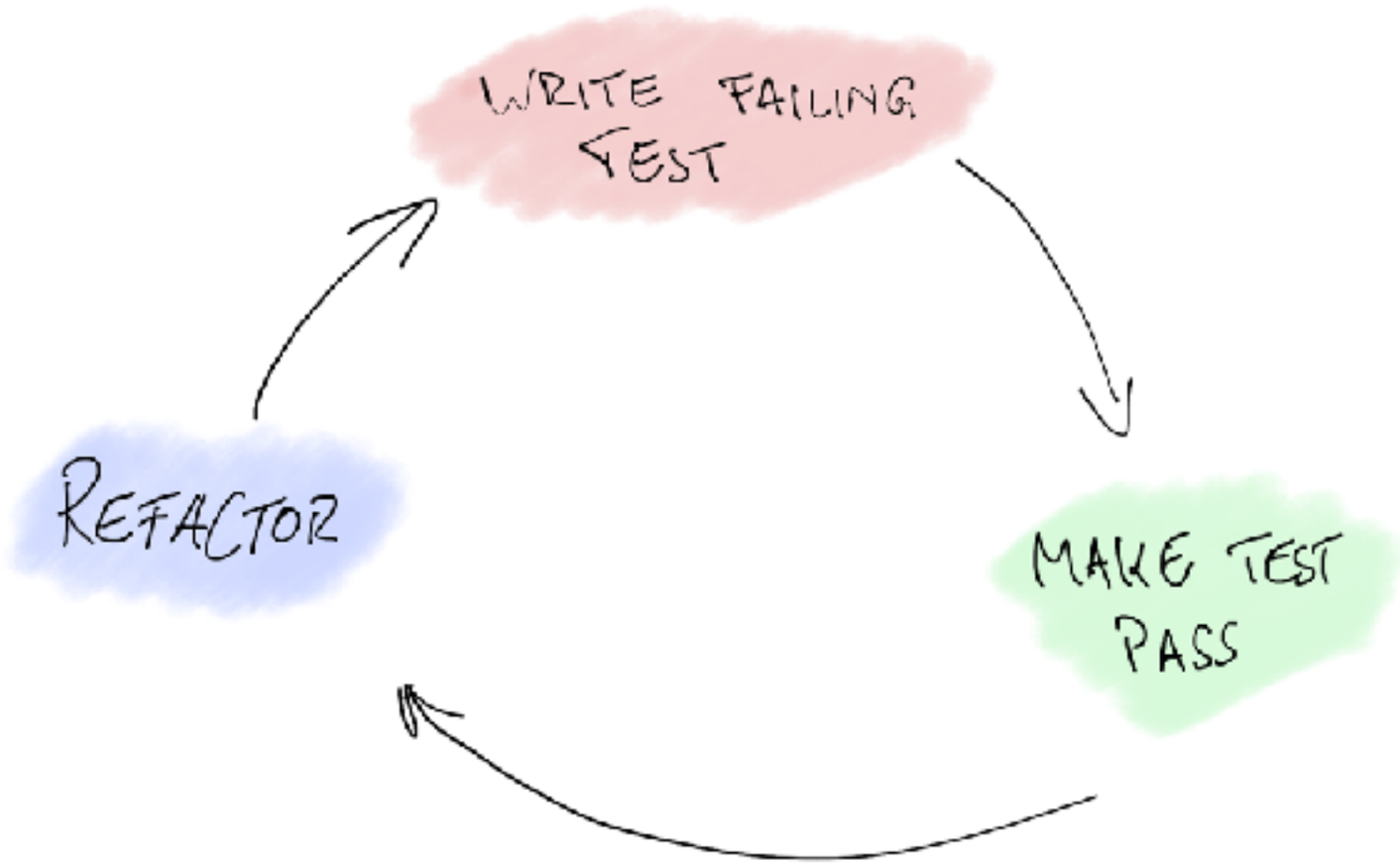
TDD DOJO

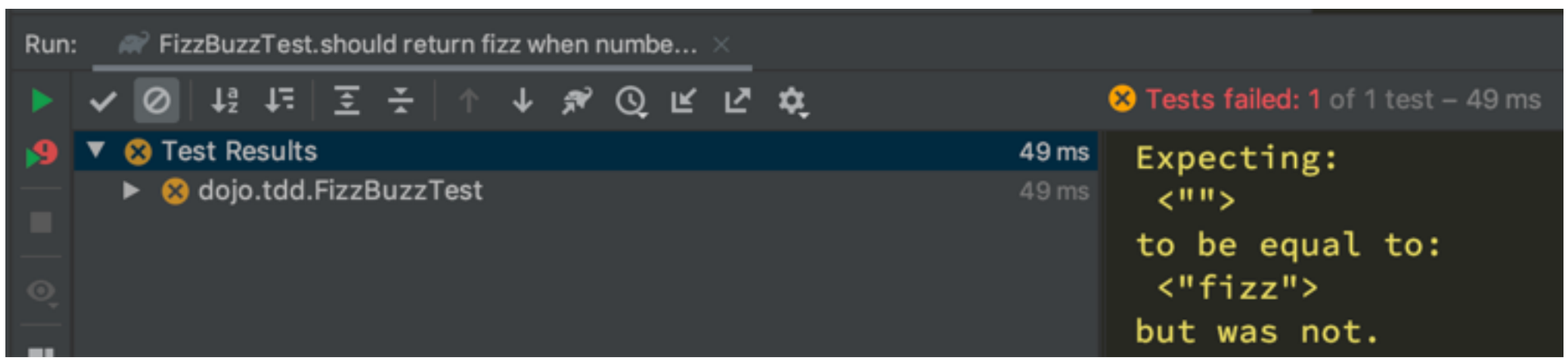

WRITE FAILING TEST
MAKE TEST PASS
REFACTOR

TDD Ping Pong

- **Step 1:** *Person 1* writes failing test
- **Step 2:** *Person 2* writes the smallest passing implementation to the test
- **Step 3:** *Person 2* writes the next test
- **Step 4:** *Person 1* writes smallest passing implementation
- **repeat:** from Step 1

Step 1 **WRITE FAILING TEST** *Person 1*

```
@Test
fun `should return fizz when number is multiple of 3`() {
    val number = 3
    val subject = FizzBuzz()

    val actual = subject.run(number)

    assertThat(actual).isEqualTo("fizz")
}
```

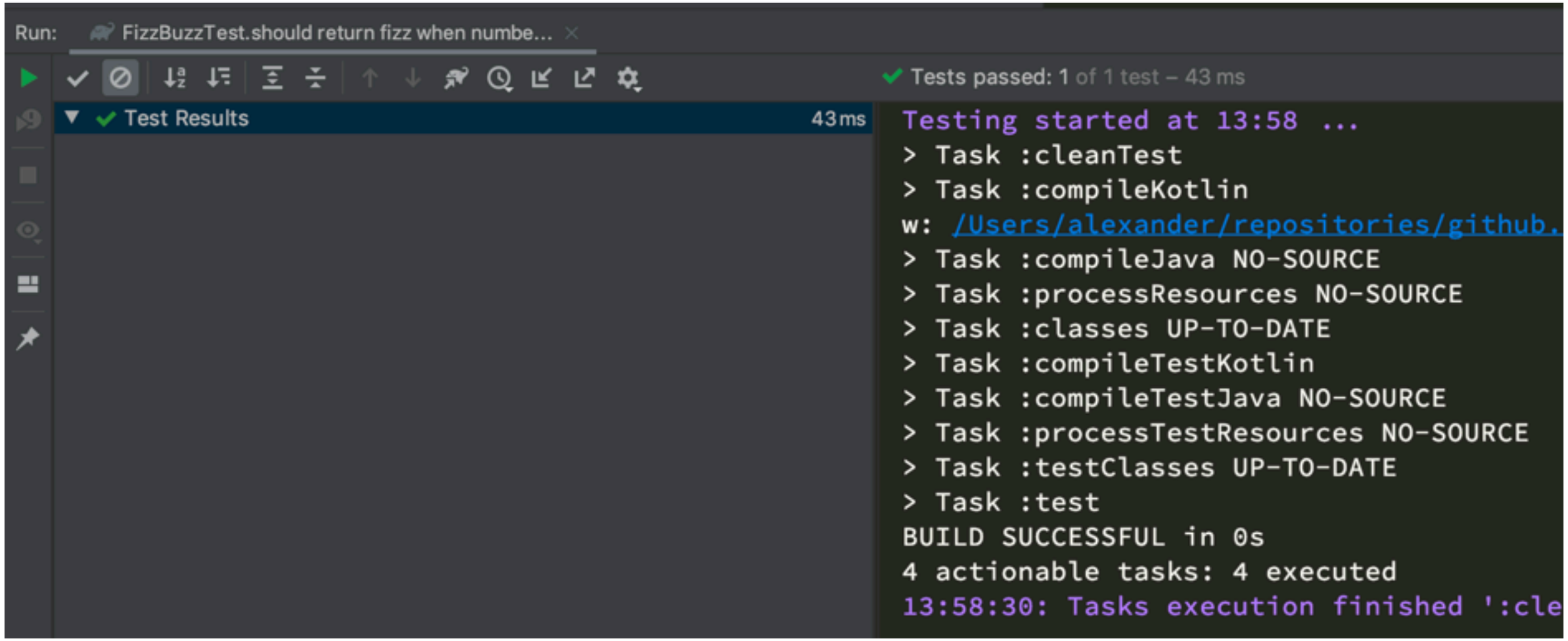


Step 2

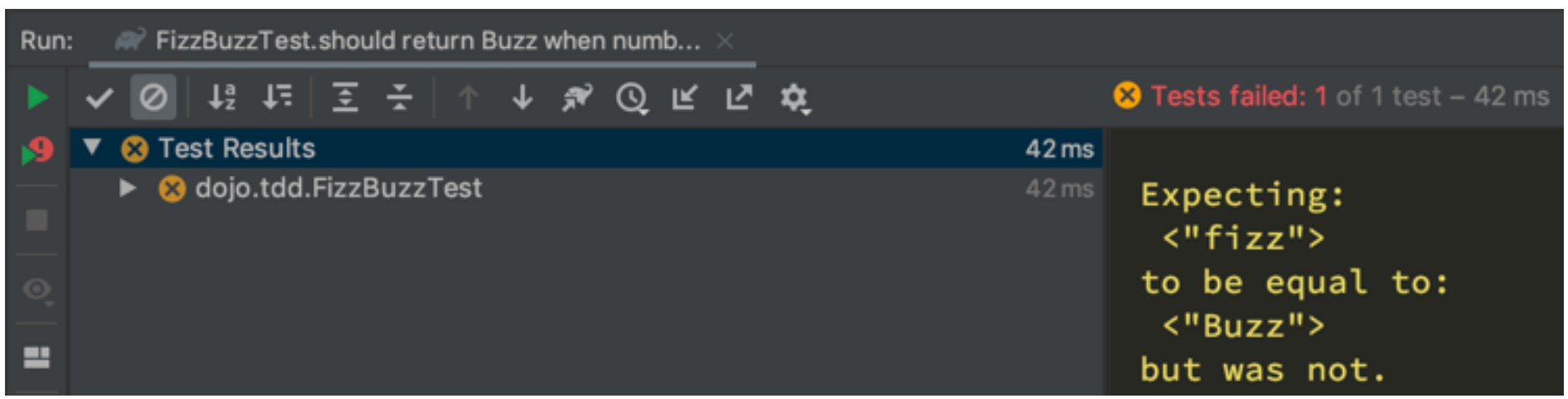
WRITE IMPLEMENTATION

Person 2

```
fun run(number: Int): String {
    return "fizz"
}
```

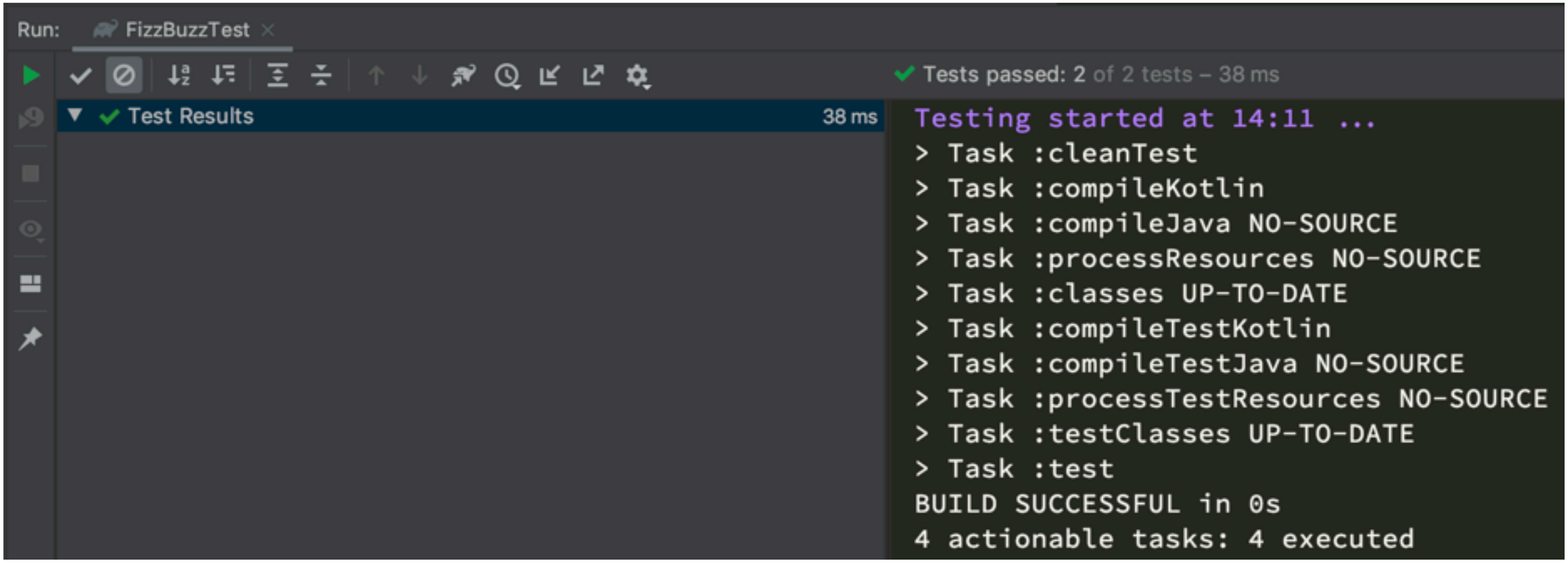


```
@Test
fun `should return Buzz when number is multiple of 5`() {
    val number = 5
    val subject = FizzBuzz()

    val actual = subject.run(number)

    assertThat(actual).isEqualTo("Buzz")
}
```



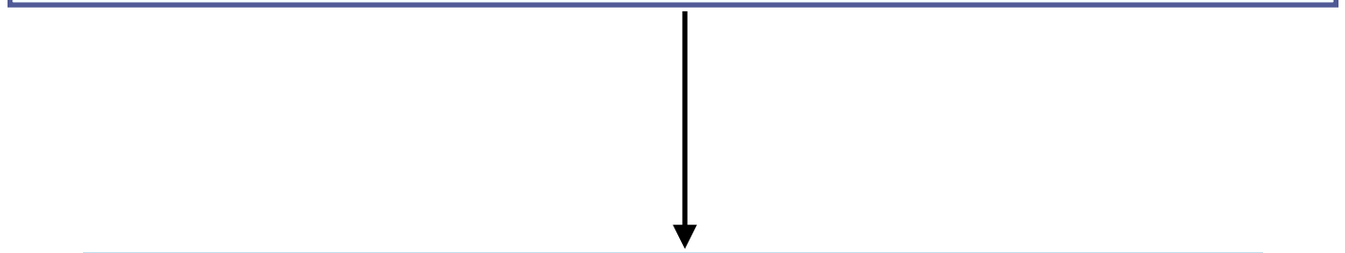
Step 4 **WRITE IMPLEMENTATION** *Person 1*

```
fun run(number: Int): String {
    return if (number % 3 == 0) "fizz"
    else if (number % 5 == 0) "Buzz"
    else ""
}
```

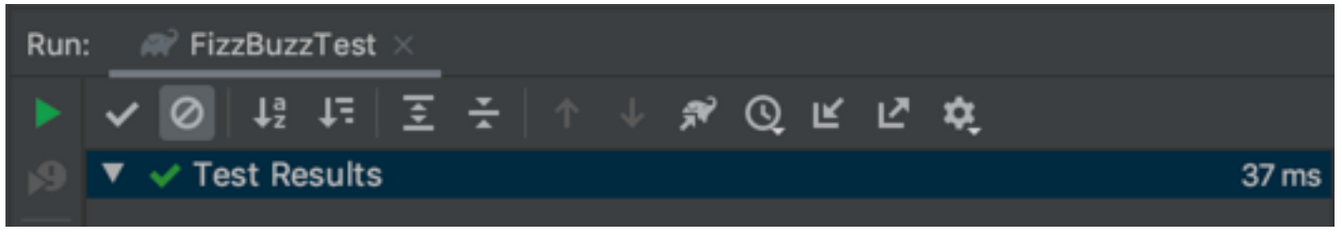


REFACTORING

```
fun run(number: Int): String {
    return if (number % 3 == 0) "fizz"
    else if (number % 5 == 0) "Buzz"
    else ""
}
```



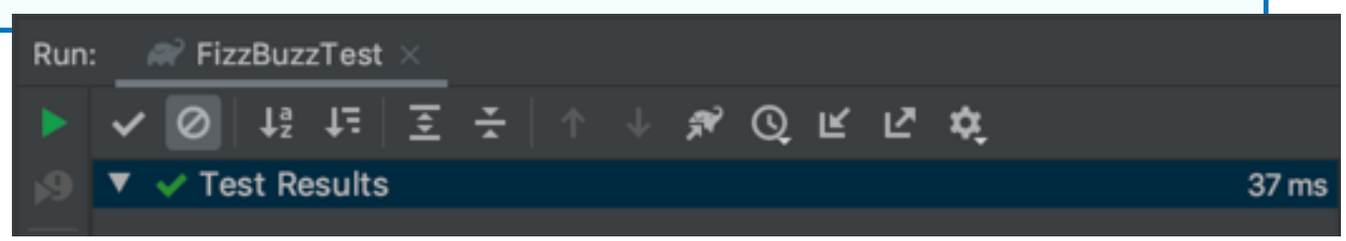
```
fun run(number: Int) = when {
        number % 3 == 0 -> "fizz"
        number % 5 == 0 -> "Buzz"
        else -> ""
    }
```



REFACTORING

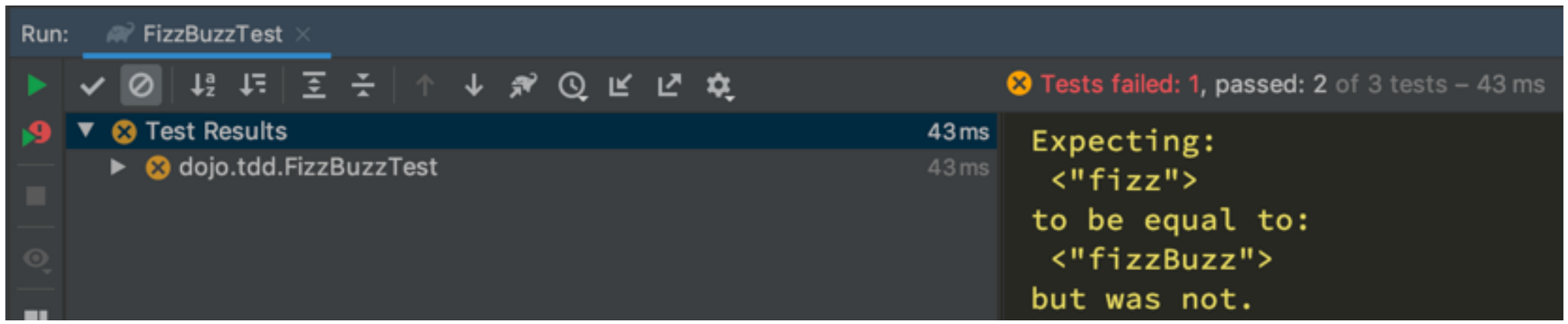
```
@Test
fun `should return fizz when number is multiple of 3`() {
    val number = 3
    val subject = FizzBuzz()

    val actual = subject.run(number)

    assertThat(actual).isEqualTo("fizz")
}
```

```
private val subject: FizzBuzz = FizzBuzz()

@Test
fun `should return fizz when number is multiple of 3`() {
    val actual = subject.run(3)

    assertThat(actual).isEqualTo("fizz")
}

@Test
fun `should return Buzz when number is multiple of 5`() {
    val actual = subject.run(5)

    assertThat(actual).isEqualTo("Buzz")
}
```



Step 1 **WRITE FAILING TEST** *Person 1*

```
@Test
fun `should return fizzBuzz when number is multiple of 5 and 3`() {
    val actual = subject.run(15)

    assertThat(actual).isEqualTo("fizzBuzz")
}
```



Step 2 **WRITE IMPLEMENTATION** *Person 2*

```
fun run(number: Int) = when {
        number % 3 == 0 && number % 5 == 0 -> "fizzBuzz"
        number % 3 == 0 -> "fizz"
        number % 5 == 0 -> "Buzz"
        else -> ""
    }
```