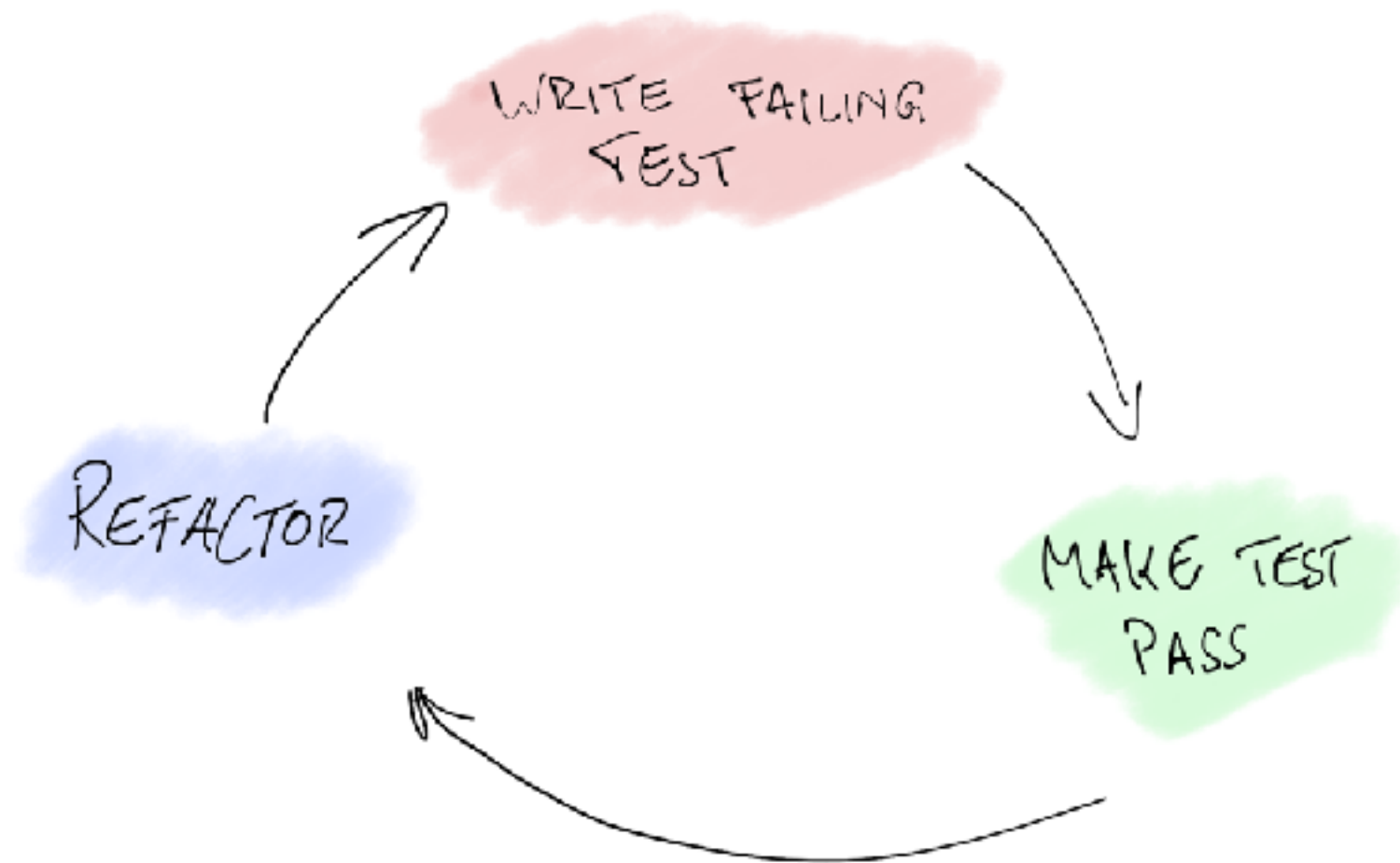




TDD DOJO



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

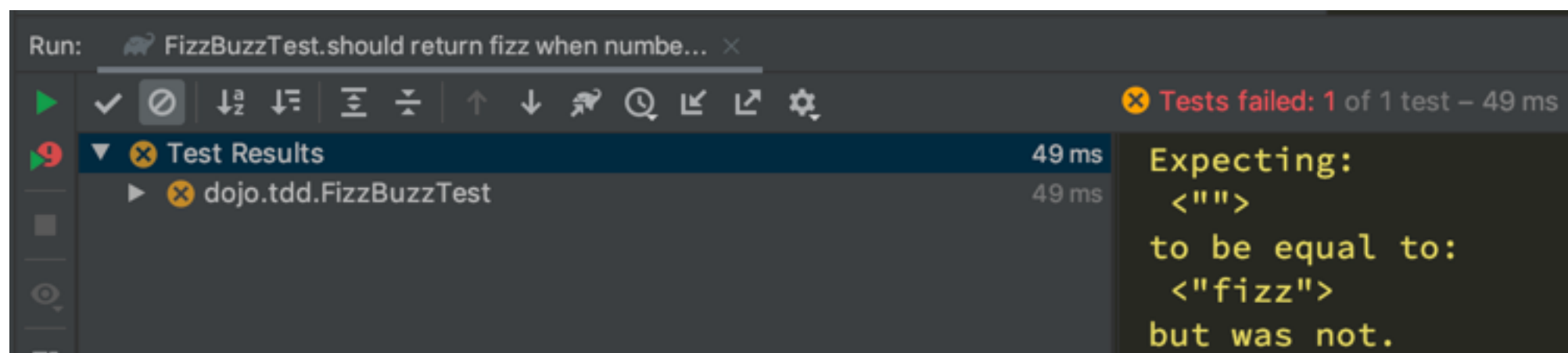
WRITE FAILING TEST

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"  
}
```

The screenshot shows an IDE interface with a 'Run' window at the top. The title bar of the 'Run' window reads 'Run: FizzBuzzTest.should return fizz when numbe...'. Below the title bar is a toolbar with icons for play, check, stop, and various sorting options. The main area of the 'Run' window is divided into two panes. The left pane, titled 'Test Results', shows a single test case 'FizzBuzzTest.should return fizz when numbe...' with a status of '43 ms' and a green checkmark. The right pane shows the build output, which includes the following text: 'Testing started at 13:58 ...', '> Task :cleanTest', '> Task :compileKotlin', 'w: /Users/alexander/repositories/github.', '> Task :compileJava NO-SOURCE', '> Task :processResources NO-SOURCE', '> Task :classes UP-TO-DATE', '> Task :compileTestKotlin', '> Task :compileTestJava NO-SOURCE', '> Task :processTestResources NO-SOURCE', '> Task :testClasses UP-TO-DATE', '> Task :test', 'BUILD SUCCESSFUL in 0s', '4 actionable tasks: 4 executed', and '13:58:30: Tasks execution finished ':cle'.

Step 3

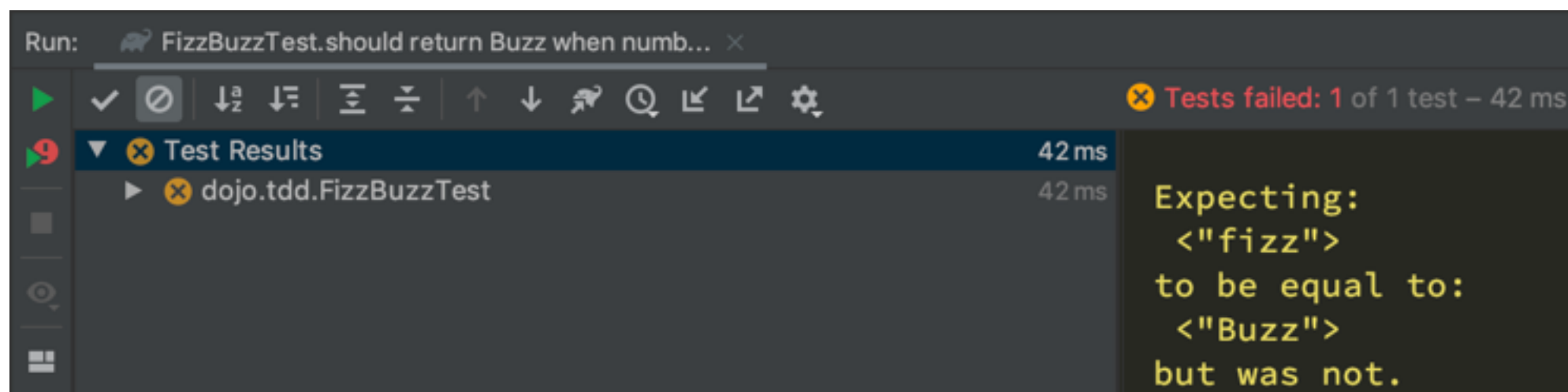
WRITE NEXT FAILING TEST

Person 2

```
@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 ""  
}
```

The screenshot shows the 'Run' window of an IDE. At the top, it says 'Run: FizzBuzzTest x'. Below this is a toolbar with various icons. The main area is divided into two panes. The left pane, titled 'Test Results', shows a green checkmark and '38 ms'. The right pane shows the build output, which includes the following text:

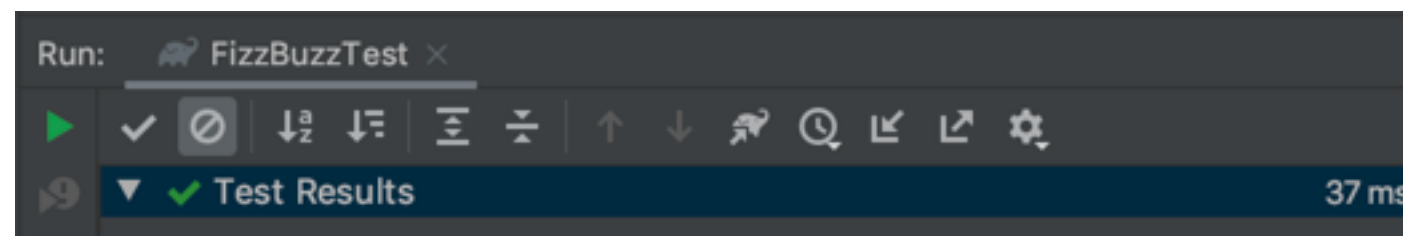
```
✓ Tests passed: 2 of 2 tests – 38 ms  
Testing started at 14:11 ...  
> Task :cleanTest  
> Task :compileKotlin  
> Task :compileJava NO-SOURCE  
> Task :processResources NO-SOURCE  
> Task :classes UP-TO-DATE  
> Task :compileTestKotlin  
> Task :compileTestJava NO-SOURCE  
> Task :processTestResources NO-SOURCE  
> Task :testClasses UP-TO-DATE  
> Task :test  
BUILD SUCCESSFUL in 0s  
4 actionable tasks: 4 executed
```

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")
}
```



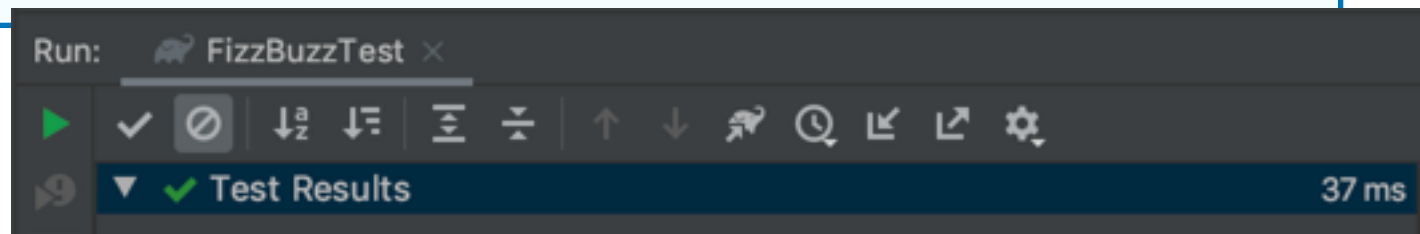
```
private val subject: FizzBuzz = FizzBuzz()

@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")
}
```



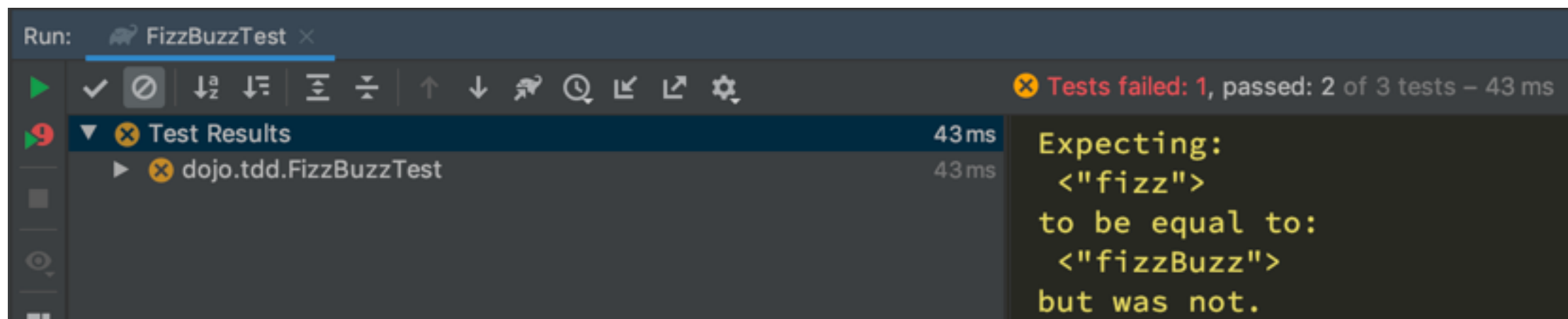
Step 1

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 -> ""  
}
```

The screenshot shows an IDE interface with a dark theme. At the top, a 'Run:' tab is active for 'FizzBuzzTest'. Below the tab is a toolbar with icons for running, debugging, and other actions. The main area is divided into two panels. The left panel, titled 'Test Results', shows a green checkmark and '37 ms'. The right panel displays the build output, starting with 'Testing started at 14:26 ...' and listing various tasks like ':cleanTest', ':compileKotlin', ':compileJava NO-SOURCE', ':processResources NO-SOURCE', ':classes UP-TO-DATE', ':compileTestKotlin', ':compileTestJava NO-SOURCE', ':processTestResources NO-SOURCE', ':testClasses UP-TO-DATE', and ':test'. The output concludes with 'BUILD SUCCESSFUL in 0s' and '4 actionable tasks: 4 executed'.

Run: FizzBuzzTest x

Tests passed: 3 of 3 tests – 37 ms

Test Results 37 ms

Testing started at 14:26 ...

- > Task :cleanTest
- > Task :compileKotlin
- > Task :compileJava NO-SOURCE
- > Task :processResources NO-SOURCE
- > Task :classes UP-TO-DATE
- > Task :compileTestKotlin
- > Task :compileTestJava NO-SOURCE
- > Task :processTestResources NO-SOURCE
- > Task :testClasses UP-TO-DATE
- > Task :test

BUILD SUCCESSFUL in 0s

4 actionable tasks: 4 executed