# Test Driven Development

JUnit 5 framework (JUnit Jupiter)

# *Test-driven development* (TDD)

- *Test-driven development* (TDD) is an approach to software development where you write tests first, then use those tests to drive the design and development of your software application.

- In TDD test cases for each functionality are created and tested first and if the test fails then the new code is written in order to pass the test and making code simple and bug-free.
- This helps to avoid duplication of code as we write a small amount of code at a time in order to pass tests.

# Test First Development

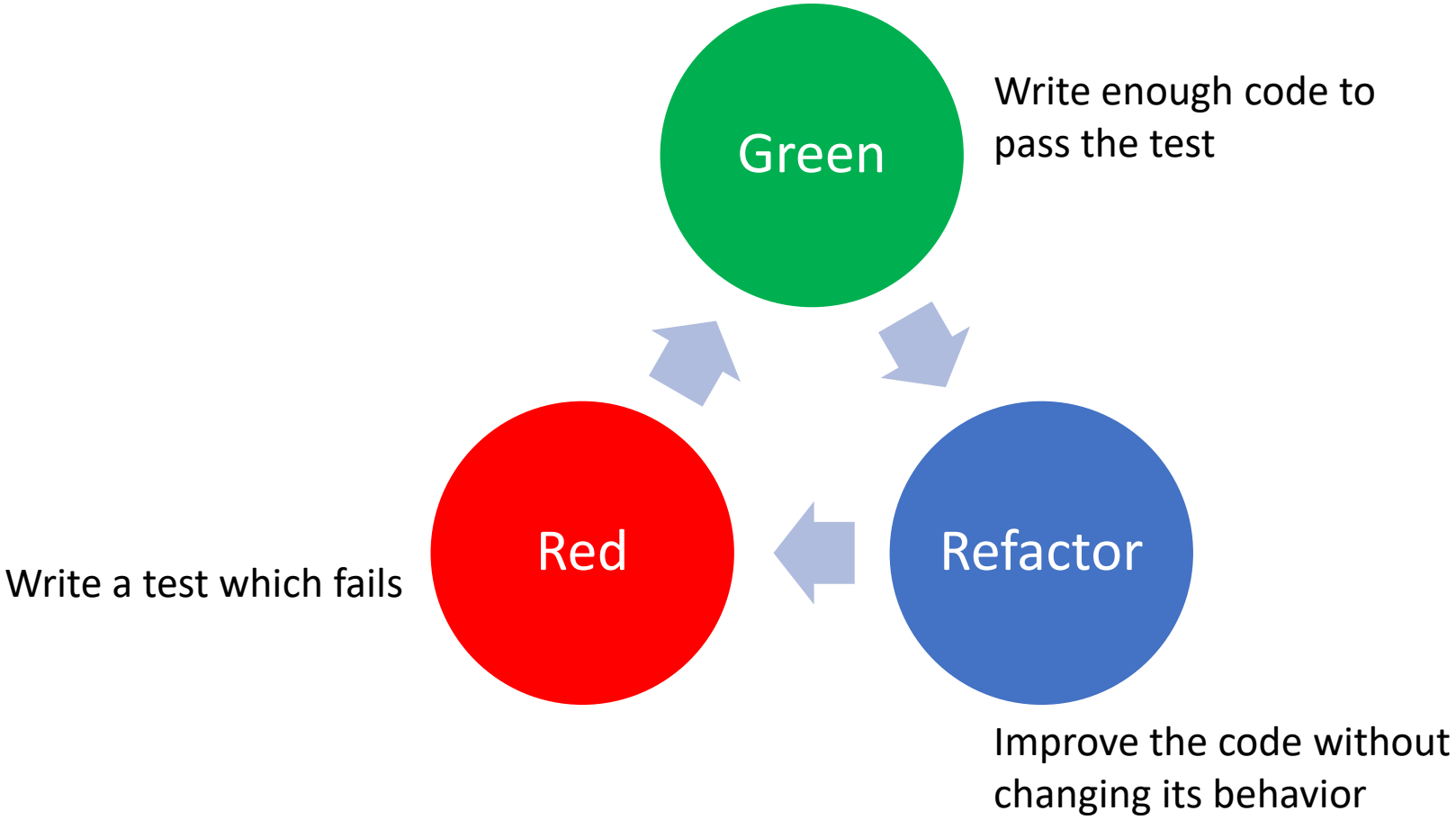
- Tests are nothing but requirement conditions that we need to test to fulfill them
- TDD ensures that your system actually meets requirements defined for it. It helps to build your confidence about your system.
- In TDD, you achieve 100% coverage test. Every single line of code is tested, unlike traditional testing.

# Red, Green, Refactor

The red, green, refactor approach helps developers compartmentalize their focus into three phases:

- Red — think about *what* you want to develop
- Green — think about *how* to make your tests pass
- Refactor — think about *how* to improve your existing implementation

# R-G-R Approach



# TDD Frameworks

A list of test driven development (TDD) frameworks

1.Junit

2.TestNG

3.csUnit and NUnit

4.Rspec

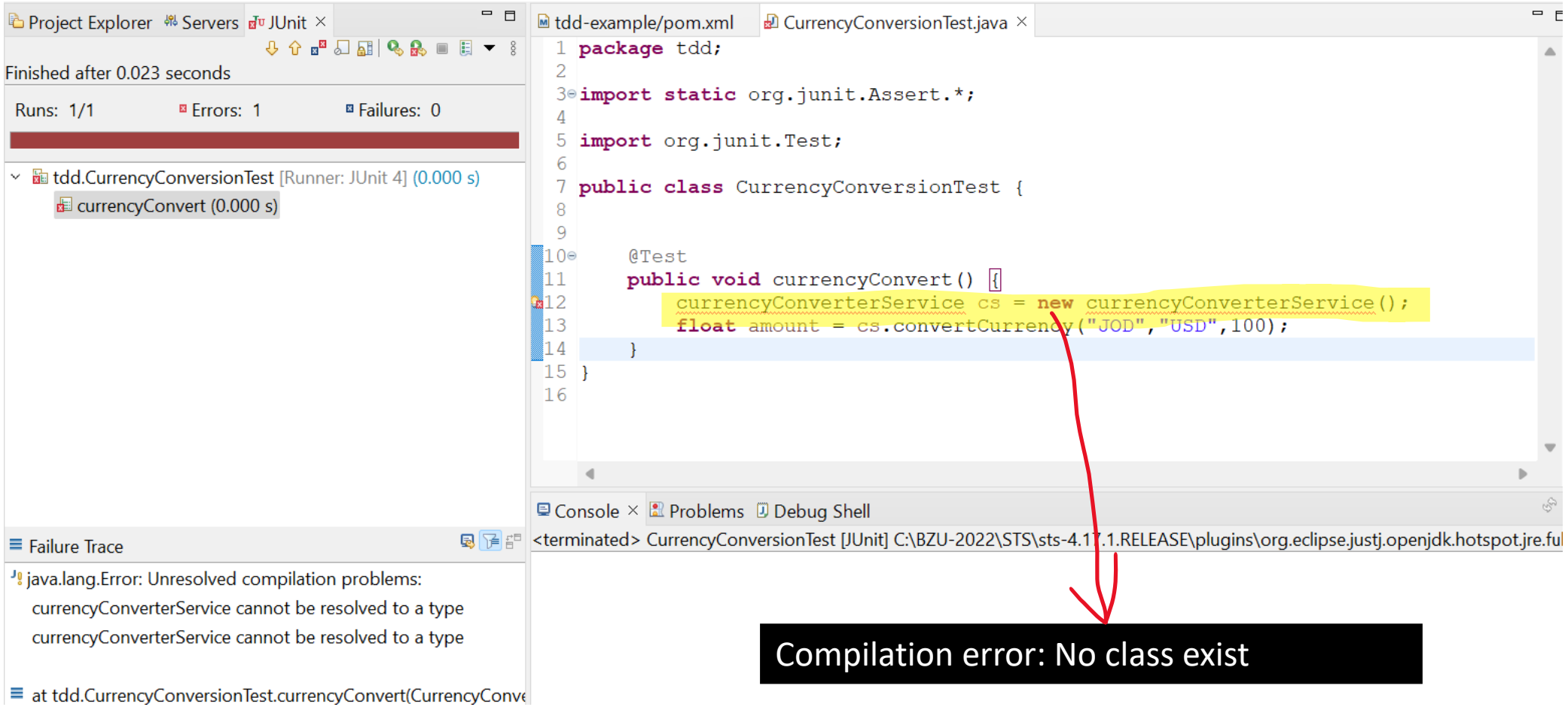
# Use Case

- Write a currency conversion service which takes two currencies (fromCurrecny, toCurrency)  and an amount then returns  the of value converted amount
- Input: fromCurrency, toCurrency, amount
- Output: amount

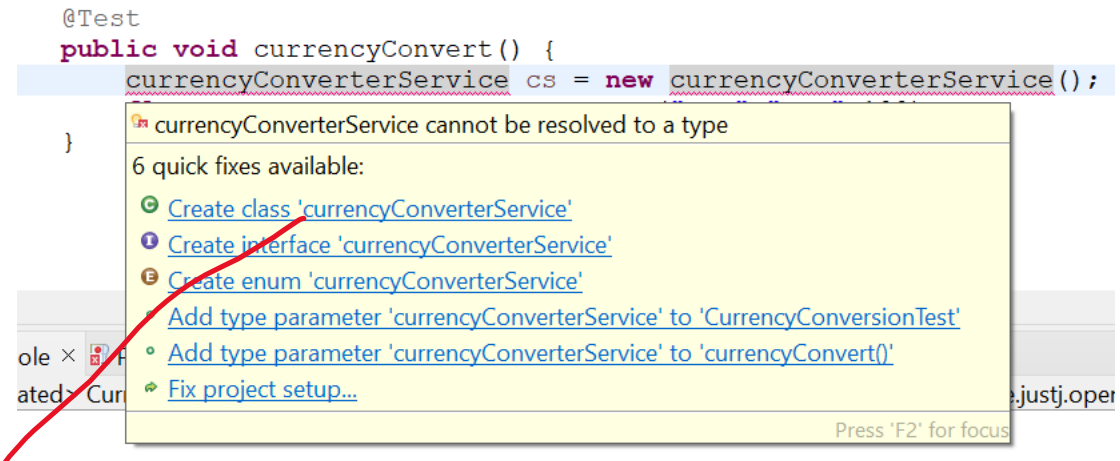
# Step#1: Create a Maven project

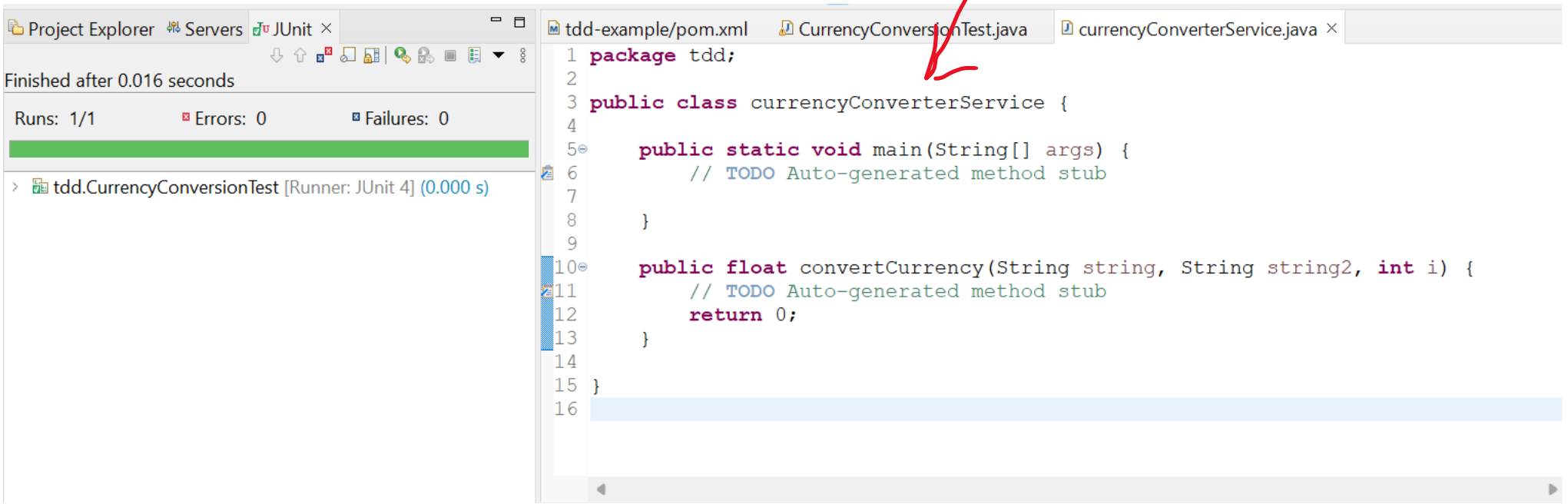
- Use junit framework

# Start Red


Project Explorer
Servers
JUnit
Finished after 0.023 seconds
Runs: 1/1
Errors: 1
Failures: 0
tdd.CurrencyConversionTest [Runner: JUnit 4] (0.000 s)
currencyConvert (0.000 s)
Failure Trace
java.lang.Error: Unresolved compilation problems:
currencyConverterService cannot be resolved to a type
currencyConverterService cannot be resolved to a type
at tdd.CurrencyConversionTest.currencyConvert(CurrencyConve
tdd-example/pom.xml
CurrencyConversionTest.java
1 package tdd;
2
3 import static org.junit.Assert.*;
4
5 import org.junit.Test;
6
7 public class CurrencyConversionTest {
8
9
10 @Test
11 public void currencyConvert() {
12 currencyConverterService cs = new currencyConverterService();
13 float amount = cs.convertCurrency("JOD","USD",100);
14 }
15 }
16
Console
Problems
Debug Shell
<terminated> CurrencyConversionTest [JUnit] C:\BZU-2022\STS\sts-4.17.1.RELEASE\plugins\org.eclipse.justj.openjdk.hotspot.jre.ful
Compilation error: No class exist

# Write Code to go Green



# Refactor

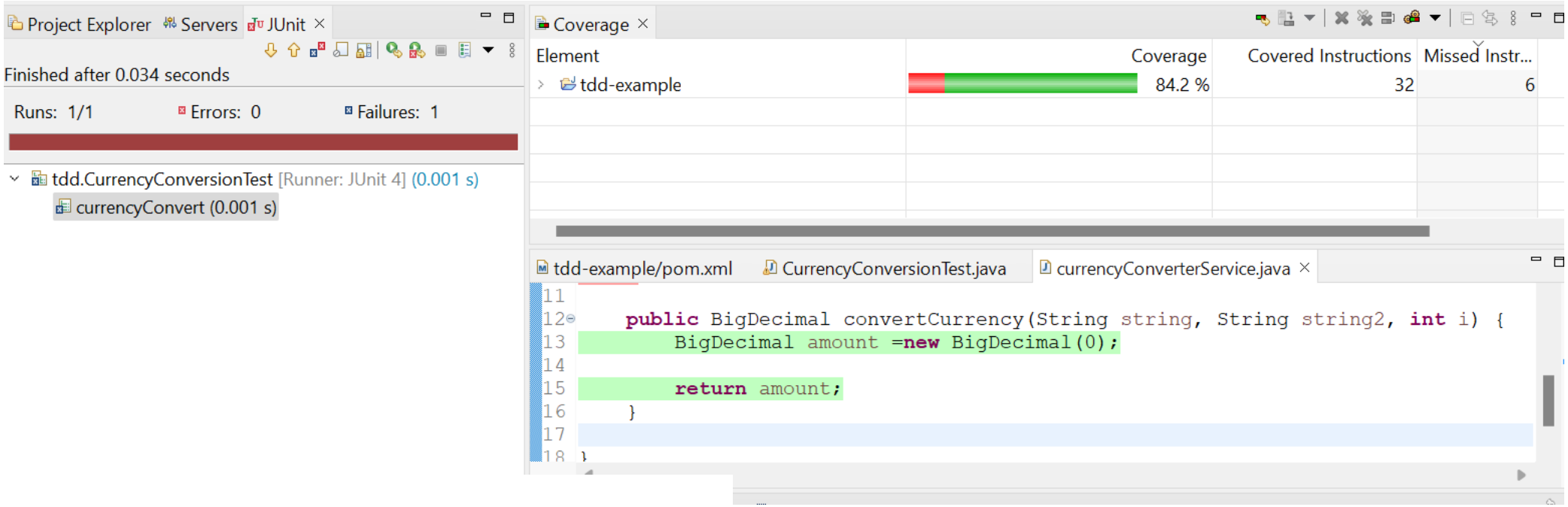
**java.math.BigDecimal vs double or float datatypes**

float and double are two primitive types, **BigDecimal is a class**. It doesn't just represent numbers but operations too. A float is a decimal numeric type ...

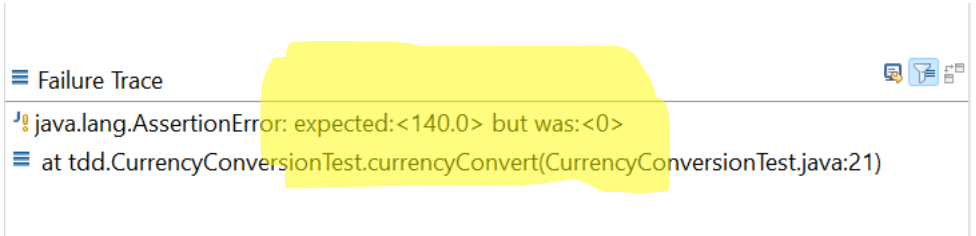
- Did we change the behavior of service?
- We are taking small steps every time!

```
public class CurrencyConversionTest {


    @Test
    public void currencyConvert() {
        currencyConverterService cs = new currencyConverterService();
        BigDecimal amount = cs.convertCurrency("JOD","USD",100);
    }
}
```

# Go Red again




```
12
13    @Test
14    public void currencyConvert() {
15        float exchangeRate = 1.4f;
16        int JOD_AMOUNT = 100;
17        float USD_AMOUNT = JOD_AMOUNT * exchangeRate;
18
19        currencyConverterService cs = new currencyConverterService();
20        BigDecimal amount = cs.convertCurrency("JOD","USD",100);
21        Assert.assertEquals(USD_AMOUNT, amount);
22    }
23 }
```

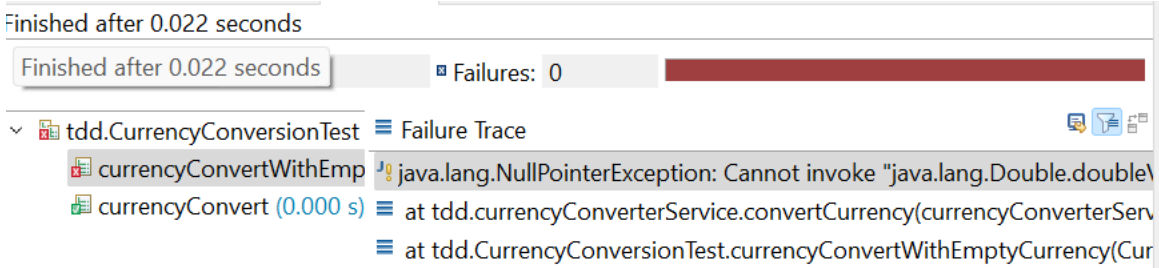


# Write Minimal Code to Green

# What can be wrong

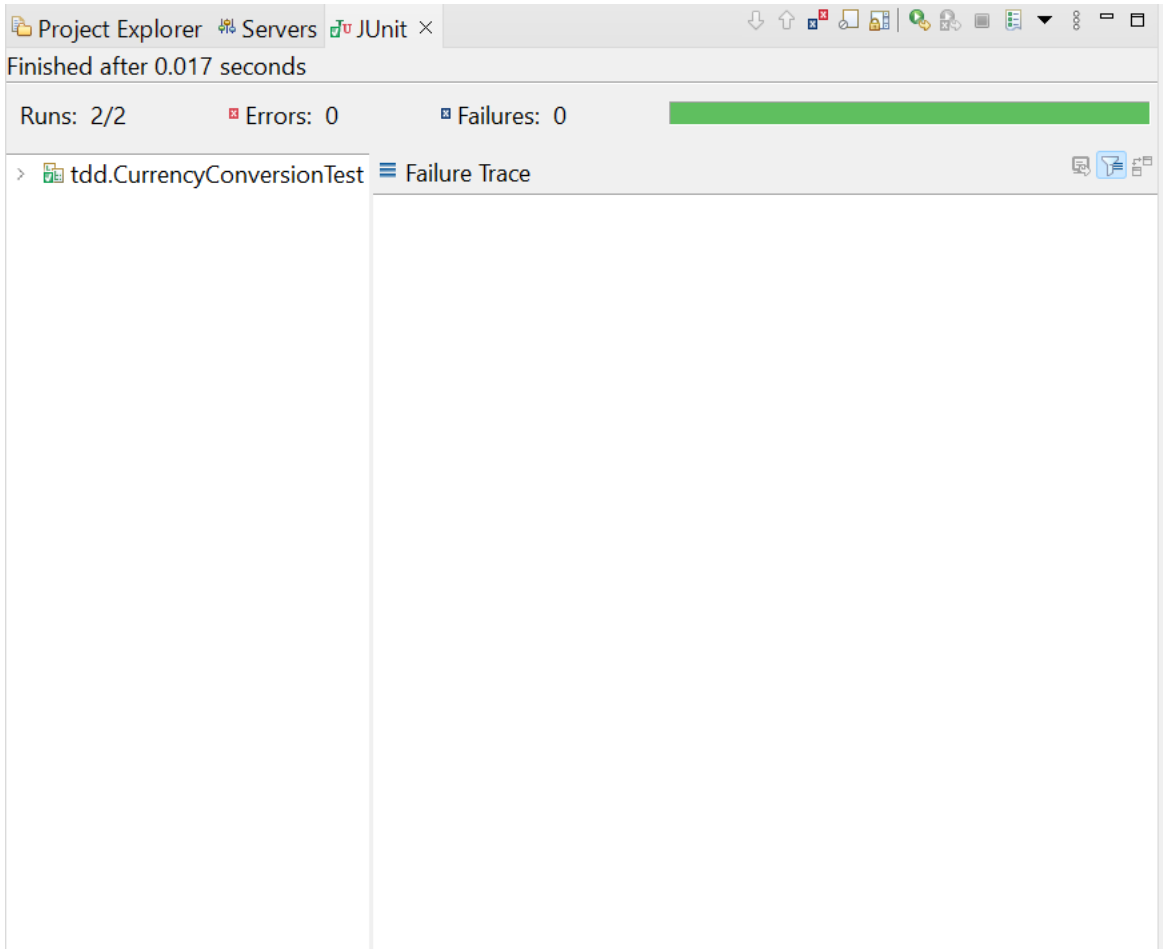
- User can pass empty currency
- User can pass a currency not in the database
- ..

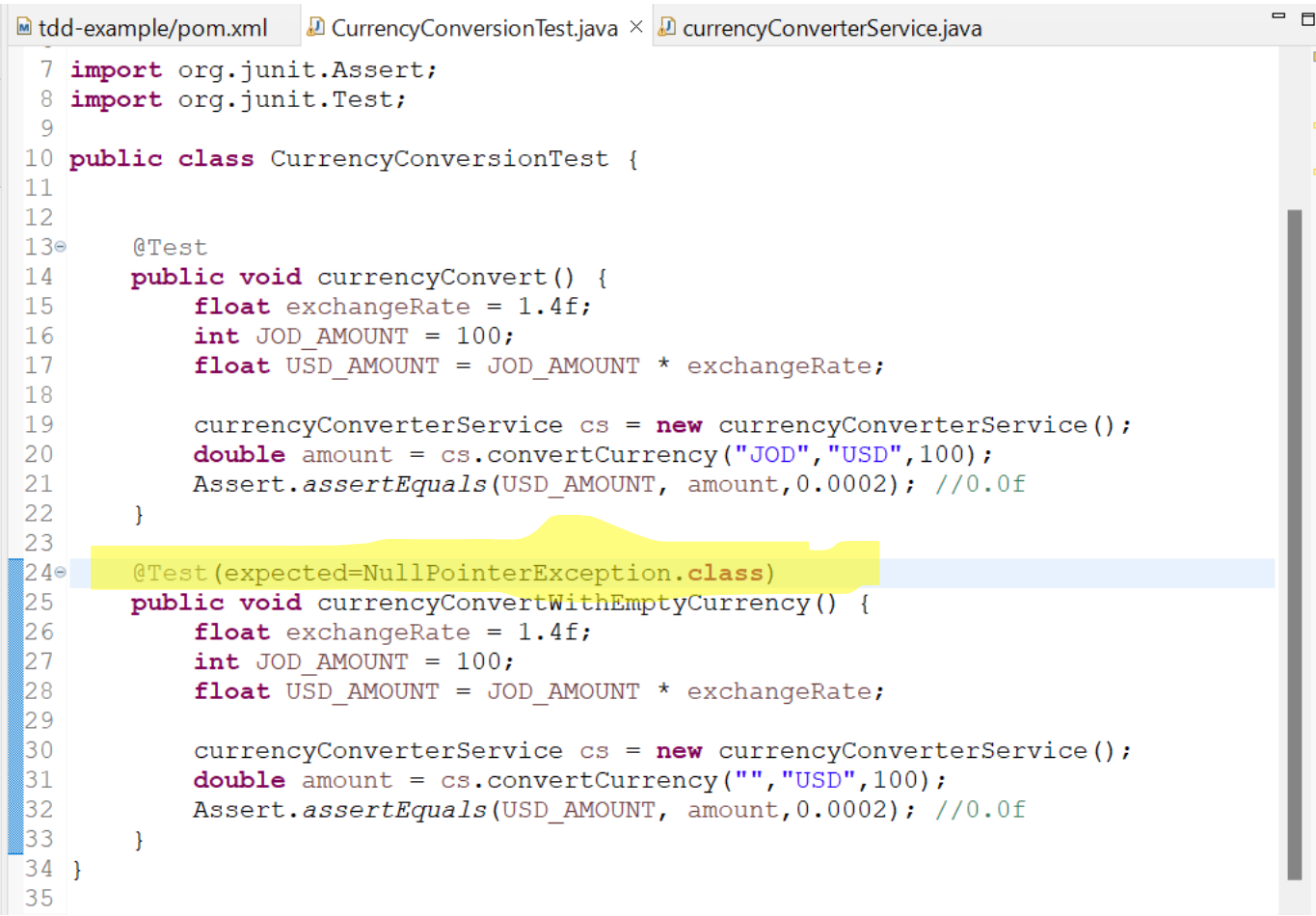
# Go Red: Empty Currency




```java
import java.math.BigDecimal;

import org.junit.Assert;
import org.junit.Test;

public class CurrencyConversionTest {


    @Test
    public void currencyConvert() {
        float exchangeRate = 1.4f;
        int JOD_AMOUNT = 100;
        float USD_AMOUNT = JOD_AMOUNT * exchangeRate;

        currencyConverterService cs = new currencyConverterService();
        double amount = cs.convertCurrency("JOD","USD",100);
        Assert.assertEquals(USD_AMOUNT, amount,0.0002); //0.0f
    }

    @Test
    public void currencyConvertWithEmptyCurrency() {
        float exchangeRate = 1.4f;
        int JOD_AMOUNT = 100;
        float USD_AMOUNT = JOD_AMOUNT * exchangeRate;

        currencyConverterService cs = new currencyConverterService();
        double amount = cs.convertCurrency("","USD",100);
        Assert.assertEquals(USD_AMOUNT, amount,0.0002); //0.0f
```

# Go Green: Empty Currency

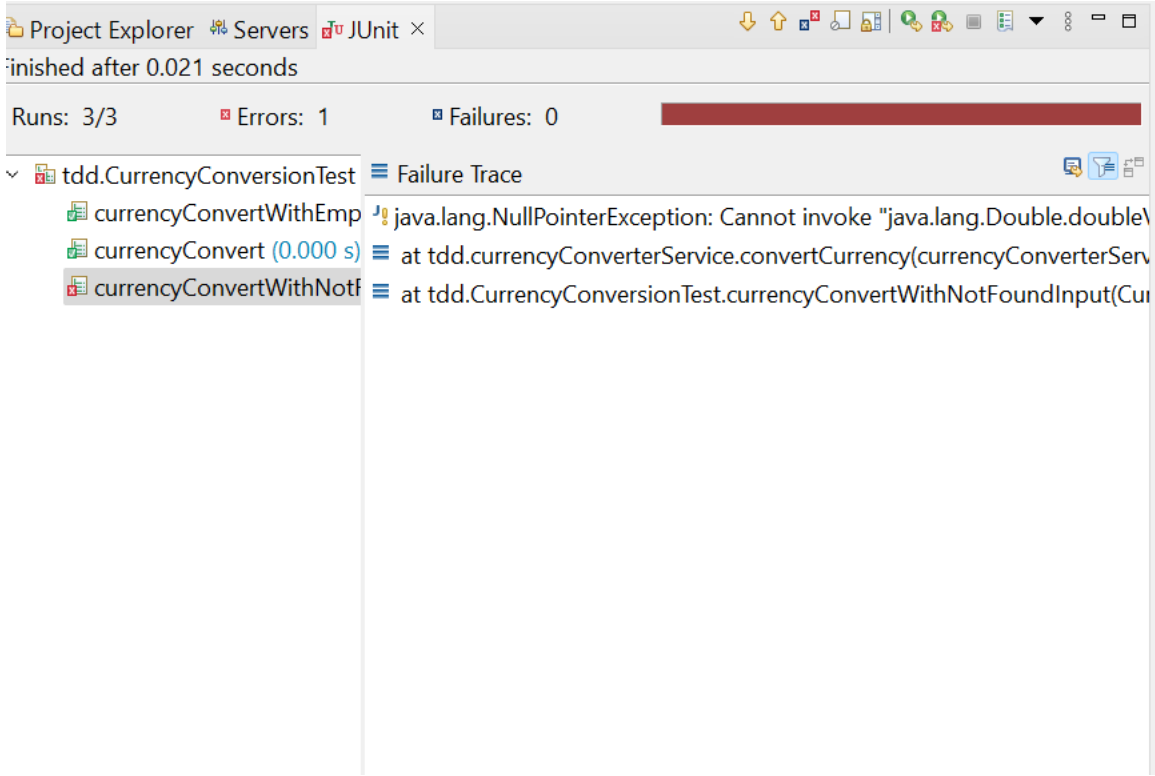


# Refactor for Empty Input

```
}

public double convertCurrency(String fromCurrency, String toCurrency, double amount) {

    if (fromCurrency.isEmpty()) {
        throw new NullPointerException("form currency cannot be empty");
    }
    Map<String, Double> rateDB = exchangeRateDB();

    double convertedAmount = rateDB.get(fromCurrency)* amount;
    return convertedAmount;
}
```

# Go Red with Not Found Currency




# Refactor

```
public double convertCurrency(String fromCurrency, String toCurrency, double amount) {
    double convertedAmount=0f;

    if (fromCurrency.isEmpty()) {
        throw new NullPointerException("form currency cannot be empty");
    }
    Map<String, Double> rateDB = exchangeRateDB();

    if (!rateDB.containsKey(fromCurrency)) {
        throw new NullPointerException("form currency is not found");
    } else
    {
     convertedAmount = rateDB.get(fromCurrency)* amount;
    }
    return convertedAmount;
}
```

# When TDD doesn’t work?