

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

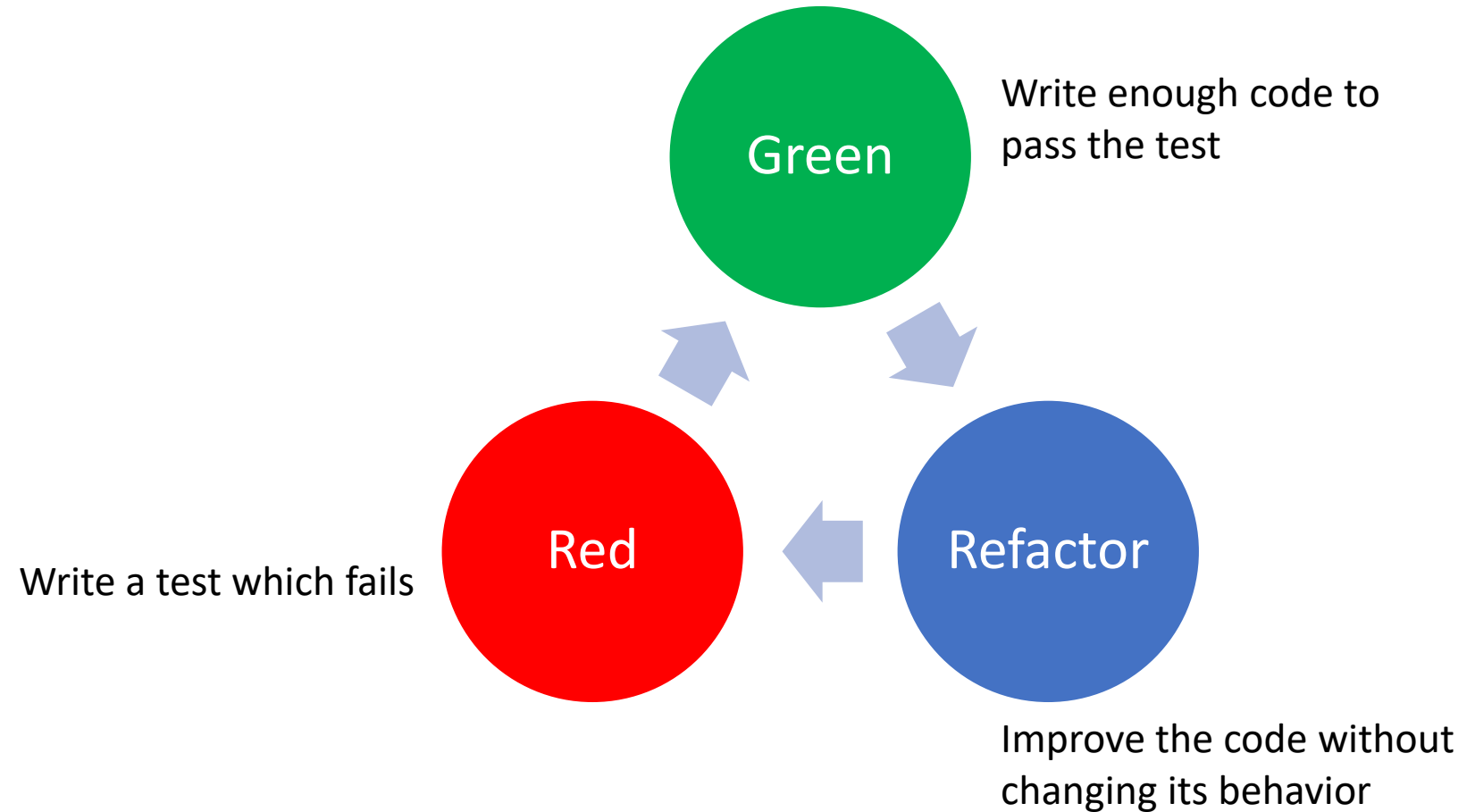
- Tests are nothing but requirement conditions that we need to test to fulfill them
- TDD ensures that your system actually meets requirements defined for it. It helps to build your confidence about your system.
- In TDD, you achieve 100% coverage test. Every single line of code is tested, unlike traditional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

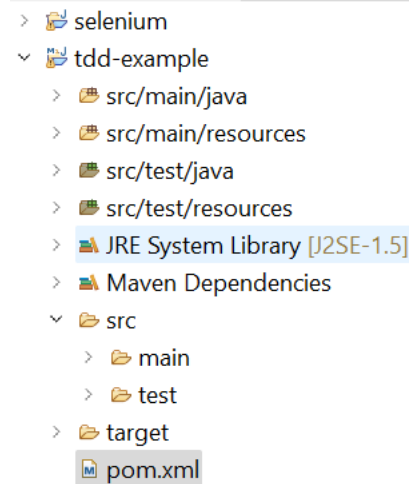
4.Rspec

Use Case

- Write a currency conversion service which takes two currencies (fromCurrency, 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



```
1<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
2  <modelVersion>4.0.0</modelVersion>
3  <groupId>birzeit.edu</groupId>
4  <artifactId>tdd-example</artifactId>
5  <version>0.0.1-SNAPSHOT</version>
6  <name>tdd-example</name>
7  <description></description>
8
9  <dependencies>
10   <dependency>
11     <groupId>junit</groupId>
12     <artifactId>junit</artifactId>
13     <version>4.13.2</version>
14     <scope>test</scope>
15   </dependency>
16 </dependencies>
17 </project>
```


Start Red

The screenshot shows the Eclipse IDE interface. On the left, the 'JUnit' tab is active, displaying a test run summary: 'Finished after 0.023 seconds', 'Runs: 1/1', 'Errors: 1', and 'Failures: 0'. Below this, a tree view shows the test class 'tdd.CurrencyConversionTest' and the method 'currencyConvert (0.000 s)'. At the bottom left, the 'Failure Trace' pane shows the error: 'java.lang.Error: Unresolved compilation problems: currencyConverterService cannot be resolved to a type'. The main editor on the right shows the 'CurrencyConversionTest.java' file. The code includes package declarations, imports for JUnit, and a test method 'currencyConvert' that instantiates 'currencyConverterService' and calls its 'convertCurrency' method. A red arrow points from the 'currencyConverterService' variable in the code to the error message. The console at the bottom right shows the command prompt for the JUnit runner.

```
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
```

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

Console × Problems × Debug Shell

<terminated> CurrencyConversionTest [JUnit] C:\BZU-2022\STS\sts-4.17.1.RELEASE\plugins\org.eclipse.justj.openjdk.hotspot.jre.fu

Compilation error: No class exist

Write Code to go Green

The screenshot shows an IDE with a test runner on the left and a code editor on the right. The test runner shows a successful run of `tdd.CurrencyConversionTest` with 0 errors and 0 failures. The code editor shows the `currencyConverterService.java` file with a red arrow pointing to the `currencyConverterService` class definition. A tooltip is visible over the `currencyConverterService` class name in the code editor, displaying the error message: "currencyConverterService cannot be resolved to a type". The tooltip also lists 6 quick fixes available:

- Create class 'currencyConverterService'
- Create interface 'currencyConverterService'
- Create enum 'currencyConverterService'
- Add type parameter 'currencyConverterService' to 'CurrencyConversionTest'
- Add type parameter 'currencyConverterService' to 'currencyConvert()'
- Fix project setup...

The code in the editor is as follows:

```
1 package tdd;
2
3 public class currencyConverterService {
4
5     public static void main(String[] args) {
6         // TODO Auto-generated method stub
7     }
8
9
10    public float convertCurrency(String string, String string2, int i) {
11        // TODO Auto-generated method stub
12        return 0;
13    }
14
15 }
16
```

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 }
```

Project Explorer Servers JUnit x

Finished after 0.034 seconds

Runs: 1/1 Errors: 0 Failures: 1

tdd.CurrencyConversionTest [Runner: JUnit 4] (0.001 s)

currencyConvert (0.001 s)

Coverage x

Element	Coverage	Covered Instructions	Missed Instr...
> tdd-example	84.2 %	32	6

tdd-example/pom.xml CurrencyConversionTest.java currencyConverterService.java x

```
11
12 public BigDecimal convertCurrency(String string, String string2, int i) {
13     BigDecimal amount = new BigDecimal(0);
14
15     return amount;
16 }
17
18 }
```

Failure Trace

```
java.lang.AssertionError: expected:<140.0> but was:<0>
at tdd.CurrencyConversionTest.currencyConvert(CurrencyConversionTest.java:21)
```

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

Finished after 0.022 seconds

Finished after 0.022 seconds

Failures: 0

Failure Trace

currencyConvertWithEmp java.lang.NullPointerException: Cannot invoke "java.lang.Double.doubleValue()" on a null object

currencyConvert (0.000 s) at tdd.currencyConverterService.convertCurrency(currencyConverterService, "JOD", "USD", 100);

at tdd.CurrencyConversionTest.currencyConvertWithEmptyCurrency(CurrencyConversionTest.java:26)

```
5 import java.math.BigDecimal;
6
7 import org.junit.Assert;
8 import org.junit.Test;
9
10 public class CurrencyConversionTest {
11
12     @Test
13     public void currencyConvert() {
14         float exchangeRate = 1.4f;
15         int JOD_AMOUNT = 100;
16         float USD_AMOUNT = JOD_AMOUNT * exchangeRate;
17
18         currencyConverterService cs = new currencyConverterService();
19         double amount = cs.convertCurrency("JOD", "USD", 100);
20         Assert.assertEquals(USD_AMOUNT, amount, 0.0002); //0.0f
21     }
22
23     @Test
24     public void currencyConvertWithEmptyCurrency() {
25         float exchangeRate = 1.4f;
26         int JOD_AMOUNT = 100;
27         float USD_AMOUNT = JOD_AMOUNT * exchangeRate;
28
29         currencyConverterService cs = new currencyConverterService();
30         double amount = cs.convertCurrency("", "USD", 100);
31         Assert.assertEquals(USD_AMOUNT, amount, 0.0002); //0.0f
32     }
33 }
```

Go Green: Empty Currency

The screenshot shows an IDE with two main panels. The left panel displays the JUnit test results for `tdd.CurrencyConversionTest`. It indicates that the test finished after 0.017 seconds, with 2/2 runs, 0 errors, and 0 failures. A green progress bar is visible. The right panel shows the source code for `currencyConverterService.java`. The code includes imports for JUnit, a test class `CurrencyConversionTest`, and two test methods. The first method, `currencyConvert()`, tests a successful conversion. The second method, `currencyConvertWithEmptyCurrency()`, is annotated with `@Test(expected=NullPointerException.class)` and tests that an empty string for the source currency results in a `NullPointerException`. The line numbers 7 through 35 are visible on the left of the code editor.

Project Explorer Servers JUnit ×
Finished after 0.017 seconds
Runs: 2/2 Errors: 0 Failures: 0
> tdd.CurrencyConversionTest Failure Trace

```
7 import org.junit.Assert;
8 import org.junit.Test;
9
10 public class CurrencyConversionTest {
11
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         double amount = cs.convertCurrency("JOD", "USD", 100);
21         Assert.assertEquals(USD_AMOUNT, amount, 0.0002); //0.0f
22     }
23
24     @Test(expected=NullPointerException.class)
25     public void currencyConvertWithEmptyCurrency() {
26         float exchangeRate = 1.4f;
27         int JOD_AMOUNT = 100;
28         float USD_AMOUNT = JOD_AMOUNT * exchangeRate;
29
30         currencyConverterService cs = new currencyConverterService();
31         double amount = cs.convertCurrency("", "USD", 100);
32         Assert.assertEquals(USD_AMOUNT, amount, 0.0002); //0.0f
33     }
34 }
35
```


Refactor for Empty Input

```
}
```

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

```
    if (fromCurrency.isEmpty()) {  
        throw new NullPointerException("from currency cannot be empty");  
    }
```

```
    Map<String, Double> rateDB = exchangeRateDB();
```

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

```
    return convertedAmount;
```

```
}
```

Go Red with Not Found Currency

The screenshot displays an IDE interface with a JUnit test run summary on the left and the source code of the failing test on the right.

JUnit Run Summary (Left Panel):

- Project Explorer: Servers
- JUnit: finished after 0.021 seconds
- Runs: 3/3
- Errors: 1
- Failures: 0
- Failure Trace:
 - tdd.CurrencyConversionTest
 - currencyConvertWithEmp: java.lang.NullPointerException: Cannot invoke "java.lang.Double.doubleValue()" on null
 - currencyConvert (0.000 s): at tdd.currencyConverterService.convertCurrency(currencyConverterService.java:31)
 - currencyConvertWithNot: at tdd.CurrencyConversionTest.currencyConvertWithNotFoundInput(CurrencyConversionTest.java:31)

Source Code (Right Panel):

The code shows the `currencyConverterService.java` file with two test methods. The first method, `currencyConvertWithEmptyInput()`, is highlighted in blue, indicating it is the current selection. The second method, `currencyConvertWithNotFoundInput()`, is also visible.

```
23  
24 @Test (expected=NullPointerException.class)  
25 public void currencyConvertWithEmptyInput() {  
26     float exchangeRate = 1.4f;  
27     int JOD_AMOUNT = 100;  
28     float USD_AMOUNT = JOD_AMOUNT * exchangeRate;  
29  
30     currencyConverterService cs = new currencyConverterService();  
31     double amount = cs.convertCurrency("", "USD", 100);  
32     Assert.assertEquals(USD_AMOUNT, amount, 0.0002); //0.0f  
33 }  
34  
35 @Test //(expected=NullPointerException.class)  
36 public void currencyConvertWithNotFoundInput() {  
37     float exchangeRate = 1.4f;  
38     int JOD_AMOUNT = 100;  
39     float USD_AMOUNT = JOD_AMOUNT * exchangeRate;  
40  
41     currencyConverterService cs = new currencyConverterService();  
42     double amount = cs.convertCurrency("YIN", "USD", 100);  
43     Assert.assertEquals(USD_AMOUNT, amount, 0.0002); //0.0f  
44 }  
45 }  
46
```

Refactor

```
public double convertCurrency(String fromCurrency, String toCurrency, double amount) {  
    double convertedAmount=0f;  
  
    if (fromCurrency.isEmpty()) {  
        throw new NullPointerException("from currency cannot be empty");  
    }  
    Map<String, Double> rateDB = exchangeRateDB();  
  
    if (!rateDB.containsKey(fromCurrency)) {  
        throw new NullPointerException("from currency is not found");  
    } else  
    {  
        convertedAmount = rateDB.get(fromCurrency) * amount;  
    }  
    return convertedAmount;  
}
```

When TDD doesn't work?