TDD: TRYING TO WALK

ATTEMPT TO IMPLEMENT THE “BOWLING KATA”

01.10.2021

Bowling Game Scoring

Write a program to calculate the score of a Ten-Pin Bowling

Input: a string representing a bowling game score
Output: the score as integer

Examples:

score string	total
X|X|X|X|X|X|X|X|X|X||XX	**300** [10 frames x 30]
9-|9-|9-|9-|9-|9-|9-|9-|9-|9-||	**90** [10 frames x 9]
5/|5/|5/|5/|5/|5/|5/|5/|5/|5/||5	**150** [10 frames x 15]
X|7/|9-|X|-8|8/|-6|X|X|X||81	**167**

Scoring Rules (short):

- Each game consists of ten “frames” with up to two tries
- If the first ball in a frame knocks down all ten pins it is called a “strike” and the frame is over (“X”)
  - Score is ten plus number of pins in next two tries
- If the second ball in a frame knocks down all ten pins it is called a “spare” (“/”)
  - Score is ten plus number of pin in next try
- If not all pins are knocked down in a frame the score is the number of pins down
- Bonus tries when last frame ends with “strike” or “spare” regarding to the scoring rules of “strike” and “spare”

START WITH A TEST

```
public void calculateScoreFromGivenGame(String game, int score) {
```

```
assertEquals(score, actualScore);
```

```
int actualScore = bowlingGameScoreCalculator.calculateScore(game);
```

```
public class BowlingGameScoreCalculatorShould {

    @ParameterizedTest
    @CsvSource({
            "'X|X|X|X|X|X|X|X|X|X||XX', 300",
            "'5/|5/|5/|5/|5/|5/|5/|5/|5/|5/||5', 150",
            "'9-|9-|9-|9-|9-|9-|9-|9-|9-||', 90"
    })
    public void calculateScoreFromGivenGame(String game, int score) {
        BowlingGameScoreCalculator bowlingGameScoreCalculator = new BowlingGameScoreCalculator();

        int actualScore = bowlingGameScoreCalculator.calculateScore(game);

        assertEquals(score, actualScore);
    }
}
```

What do we want to test?

- We want to test behaviour
  - BowlingGameScoreCalculatorShould…
  - …calculateScoreFromGivenGame
- Assert that the calculated score is the expected from the example(s)

What shall the software do?

- Calculate the score for a given game
  - The behaviour is the API
  - 1 public interface with input «game» as String and «score» as Integer

- We can make one parameterized test, because for every example the score calculating rules stay the same
- We did not write any code but looking at the requirements we test any behaviour the program should have

WHAT DO WE ACHIEVE FROM THE TEST?

- **Clear “given”, “when”, “then” and it is readable like simple text:**
  - BowlingGameScoreCalculatorShould…
    - …calculateScoreFrom**Given**Game
      - **Given** a played game
      - **When** called to calculate
      - **Then** return the (correct) score

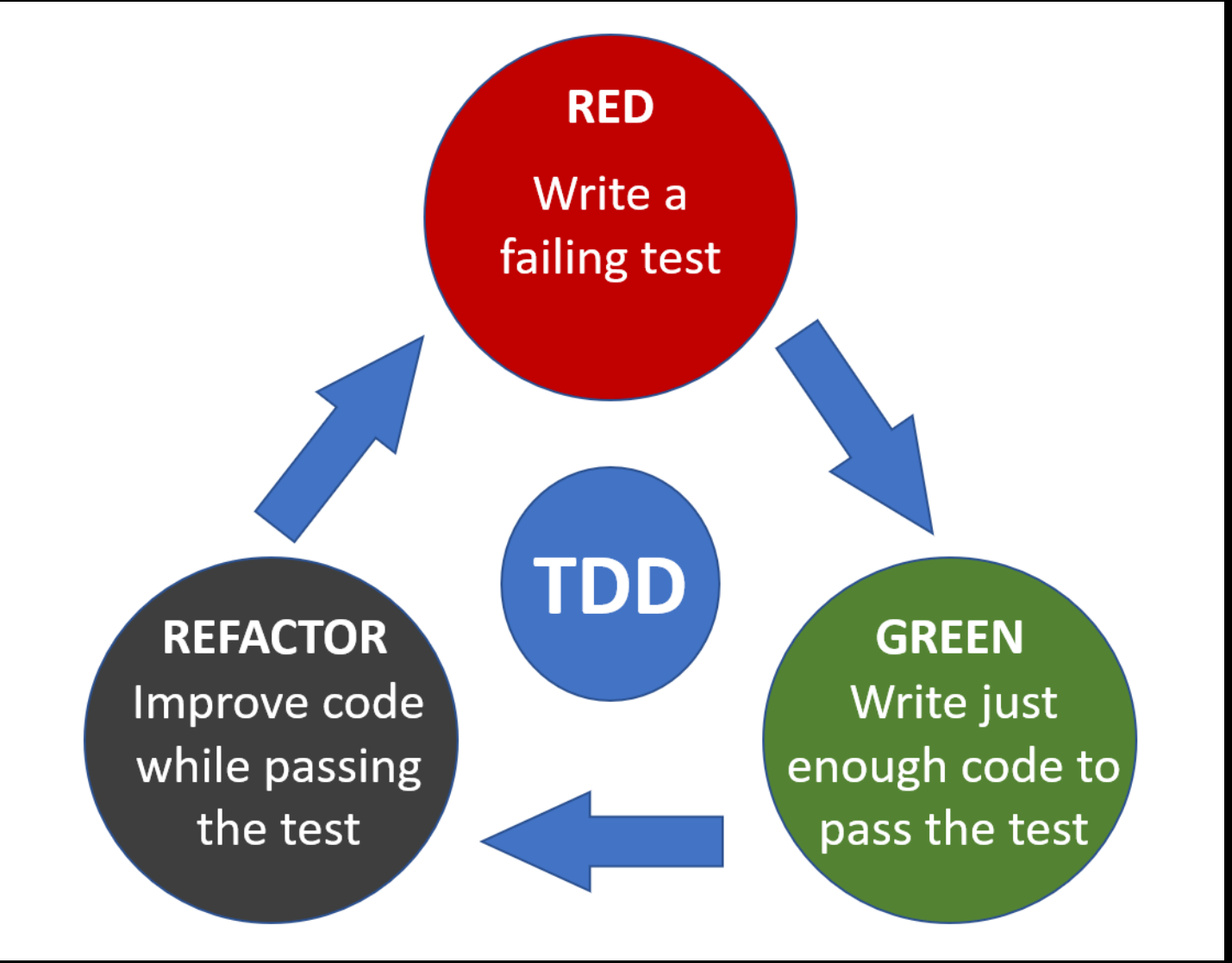
- **When the implementation changes…**
  - Example: the rules of calculating the score change
    - ➢ The test does not have to change!
    - ➢ If the API does not change, the caller does not change
  - Example: using an extern library instead of our code
    - ➢ Just the call to production code and maybe the input changes
    - ➢ The structure of the test does not have to change!

```
@ParameterizedTest
@CsvSource({
        "'X*X*X*X*X*X*X*X*X*X**XX', 300",
        "'5/*5/*5/*5/*5/*5/*5/*5/*5/*5/**5', 150",
        "'9-*9-*9-*9-*9-*9-*9-*9-*9-*9-**', 90",
        "'X*7/*9-*X*-8*8/*-6*X*X*X**81', 167"
})
public void calculateScoreFromGivenGame(String game, int score) {
    ExternBowlingScoreCalculator bowlingGameScoreCalculator = new ExternBowlingScoreCalculator();

    int actualScore = bowlingGameScoreCalculator.calculateScore(game);

    assertEquals(score, actualScore);
}
```

CHECKING FOR TEST SMELLS

• Not testing anything	-> testing the score calculation
• Test too many things at once	-> only testing the score calculation
• Too many assertions	-> only one
• Assertions mixed with actions	-> only one action seperated from one assertion
• Testing or containing irrelevant information	-> only one object is created and one method is called
• Conditional test logic	-> no conditions in test
• Test too long	-> just one line for given, when, then
• Excessive setup	-> no setup necessary but the instance creation for calling the calculator

HOW TO GO ON?


RED
Write a failing test
GREEN
Write just enough code to pass the test
REFACTOR
Improve code while passing the test
TDD

WRITE A RED TEST

```
"'X*X*X*X*X*X*X*X*X*X**XX', 300",
```

```
public class BowlingGameScoreCalculator {
    public int calculateScore(String game) {
        return 0;
    }
}
```

```
org.opentest4j.AssertionFailedError:
Expected :300
Actual   :0
```

Introduce first example

Make the test fail for the right reason

- Create BowlingGameScoreCalculator and method calculateScore
  - Return 0 to make it fail for the right reason (no implementation)

MAKE THE TEST GREEN

```java
public class BowlingGameScoreCalculator {
    public int calculateScore(String game) {
        return 300;
    }
}
```

```java
public class BowlingGameScoreCalculator {
    public int calculateScore(String game) {
        int score = 0;
        String[] frames = game.split( regex: "\\|");
        for(String frame : frames) {
            if ("X".equals(frame)) {
                score += 30; // this is actually wrong, only in a full strike game true
            }
        }
        return score;
    }
}
```

Write just enough code to make the test green

- Fake implementation

- Obvious implementation
  - Split the input String at the “|”
  - Count the X
  - Because we know every one of the 10 frames will be scored 30, we can add 30 for every frame
  - Due to the delimiter “|” the extra rolls are ignored

REFACTORING

OBJECT CALISTHENICS RULES

- Wrap all primitives and Strings
  - “X” (and later “/” or “-”) can be wrapped in Enum or Class
  - String-Array “frames” to Collection of type “Frame”
  - Wrap Delimiter “|” and “||” for the extra tries?
- Only one level of indentation
  - If-condition can move to “Frame”
    - We get a method call for the condition, ok? (-> TPP)

TRANSFORMATION PRIORITY PREMISE

- When moving condition to “Frame”-class
  - Condition: 6; Function: 12
- *Does this make it worse?*

TRIANGULATION INSTEAD OF REFACTORING

```java
public class BowlingGameScoreCalculator {
    public int calculateScore(String game) {
        int score = 0;
        String[] frames = game.split( regex: "\\|");
        for(String frame : frames) {
            if ("X".equals(frame)) {
                score += 30;
            } else if (frame.contains("/")) {
                score += 15;
            } else {
                char[] singleRolls = frame.toCharArray();
                for (char roll : singleRolls) {
                    if (Character.isDigit(roll)) {
                        score += (int) roll-'0';
                    }
                }
            }
        }
        return score;
    }
}
```

This is what can happen after triangulate the next 2 examples without refactoring :D

- Object Calesthenics Rules
  - Don’t use ELSE keyword
- TPP:
  - 2 **conditions(6)** in **loop(10)** and a **nested loop(10)** in ELSE with **another condition(6)** in it
    - Can’t even calculate the TPP score easily @.@
- Refactor it!!!

AFTER REFACTORING

```java
public class BowlingGameScoreCalculator {
    private final Frames frames = new Frames();
    public int calculateScore(String gameResult) {
        int score = 0;
        Frames.buildFrames(gameResult);

        for (Frame frame : frames.getFrames()) {
            score = frame.incrementScore(score, frames);
        }
        return score;
    }
}
```

“Frames” is parsing the String input

- Calculator does not parse, just calculate final score

“Frame” has the responsibility to calculate the score for every single frame

- Logic of what is “strike” and so on is capsuled in “Frame”
  - Frame-Logic is in Frame-Domain

Actual number of frames played (10) does not have to be known

- works by rule change (e.g. 15 frames per game)

Logic of additional tries (strikes, spares) is capsuled in “Frame “

- “Frame” can be replaced by any other unit
  - Test does not care

QUESTIONS (FROM ME)

Is the TPP score too high?

- Scalar (int score) -> 4
- Function (buildFrames, incrementScore) -> 24
- Mutation (score = …) -> 13
- Loop (Frame frames : frames) -> 10

- Does this multiply the score of function and mutation?

```
public class BowlingGameScoreCalculator {
    private final Frames frames = new Frames();
    public int calculateScore(String gameResult) {
        int score = 0;
        Frames.buildFrames(gameResult);

        for (Frame frame : frames.getFrames()) {
            score = frame.incrementScore(score, frames);
        }
        return score;
    }
}
```

MERCI

Looking forward to next module
And
Have a great weekend!

Let’s practice and become competent in what we are doing!