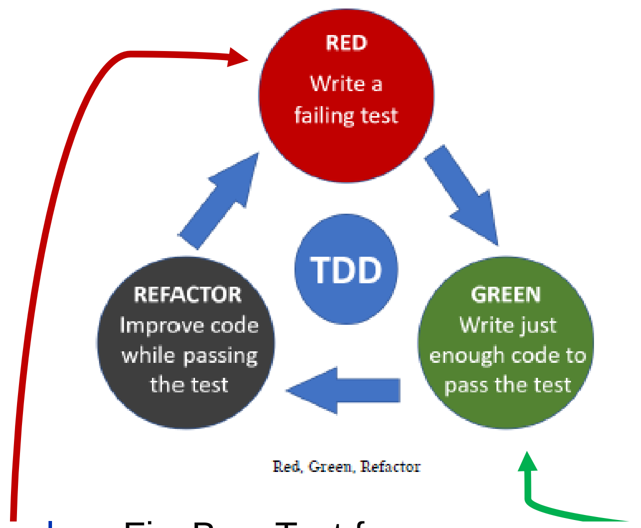
Red – Green – Refactor
without a given interface

Luzern, 1. November 2021

ALCOR Academy Training

Motivation – classic TDD cycle with a given interface


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

Red, Green, Refactor

FizzBuzz

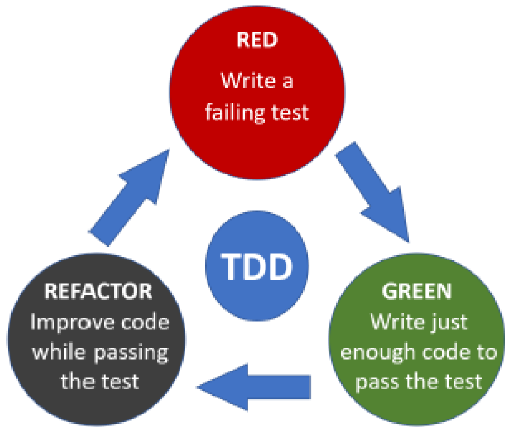
*Write a function that **takes numbers** from 1 to 100 and **outputs** them as a **string**.*
*For **multiples of three** returns **Fizz** instead of the number and for the **multiples of five** returns **Buzz**.*
*For numbers which are **multiples of both** three and five returns **FizzBuzz**.*

```
class FizzBuzzTest {

    @ParameterizedTest
    @CsvSource({
            "1, '1'", "3, 'Fizz'", "5, 'Buzz'", "15, 'FizzBuzz'"
    })
    public void calculateFizzBuzzForGivenNumber(int number,
String
            expectedResult) {
        FizzBuzz fizzbuzz = new FizzBuzz();
        String actualResult = fizzbuzz.calculate(number);
        assertEquals(expectedResult, actualResult);
    }
}
```

```
ıblic cla

public

    retu
}
```

```
public class FizzBuzz {

    public String calculate(int number) {
        String result = "";
        if (isMultipleOf(number, 3)) {
            result = result + "Fizz";
        }
        if(isMultipleOf(number, 5)) {
            result = result + "Buzz";
        }
        return result.equals("") ? String.valueOf(number) :
result;
    }

    private boolean isMultipleOf(int dividend, int divisor) {
        return dividend % divisor == 0;
    }
}
```

Motivation – classic TDD cycle without a given interface


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

Red, Green, Refactor

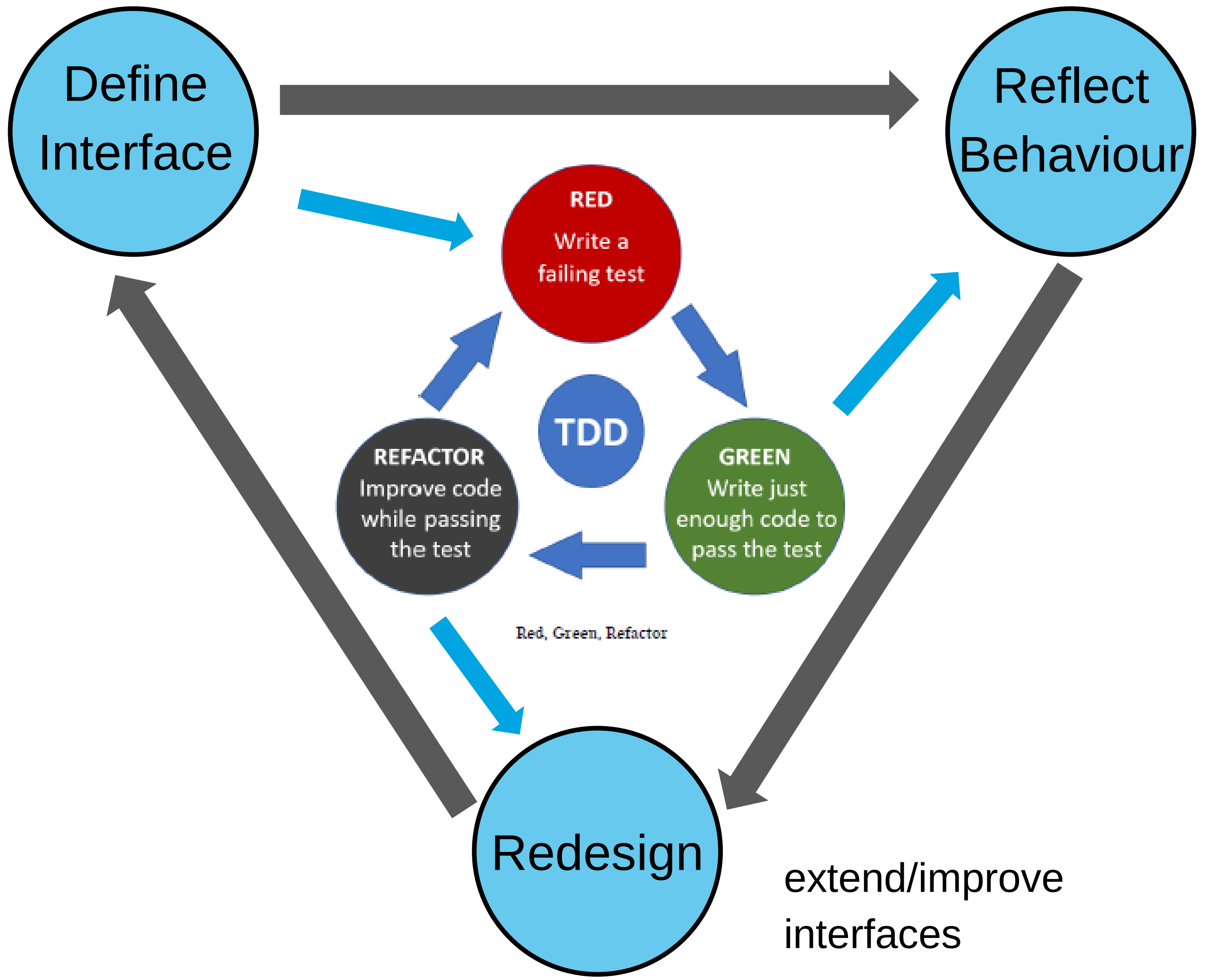
TicTacToe

- *X always goes first.*
- *Players alternate placing X's and O's on the board.*
- *Players cannot play on a played position.*
- *A player with three X's or O's in a row (horizontally, vertically, diagonally) wins.*
- *If all nine squares are filled and neither player achieves three in a row, the game is a draw.*

- Interface is unclear !
- How to start for the first RED test ?
- Which requirement has priority ?
- …

```
class TicTacToeGameShould {
    @Test
    void doSomething() {
        assertEquals(true, true);
    }
}
```

Idea – Extending classic TDD cycle



Identify behaviour

did we test the right thing ?

extend/improve interfaces

TicTacToe – first try

Interfaces	Tests
boolean putStone(int coordinate);	puttingAStoneDoesntFinishTheGame finishTheGameAfterPuttingNineStones notFinishTheGameAfterPuttingTwoStones
String giveNextPlayer();	makeXtheFirstPlayer makeOtheSecondPlayer makeThePlayersAlternate
String getWinner();	makeOWinOnTopRow makeXWinOnTopRow

Command-Query-Responsibility-Segregation

behavior ?
relevance
priority

- *X always goes first.*
- *Players alternate placing X's and O's on the board.*
- *Players cannot play on a played position.*
- *A player with three X's or O's in a row (horizontally, vertically, diagonally) wins.*
- *If all nine squares are filled and neither player achieves three in a row, the game is a draw.*

TicTacToe – second try

Interfaces

- Player getCurrentPlayer()
- void placeStone(Tile tile)
- Player getWinner()

Tests

makeXTheFirstPlayer

makeOTheSecondPlayer
makeThePlayersAlternate

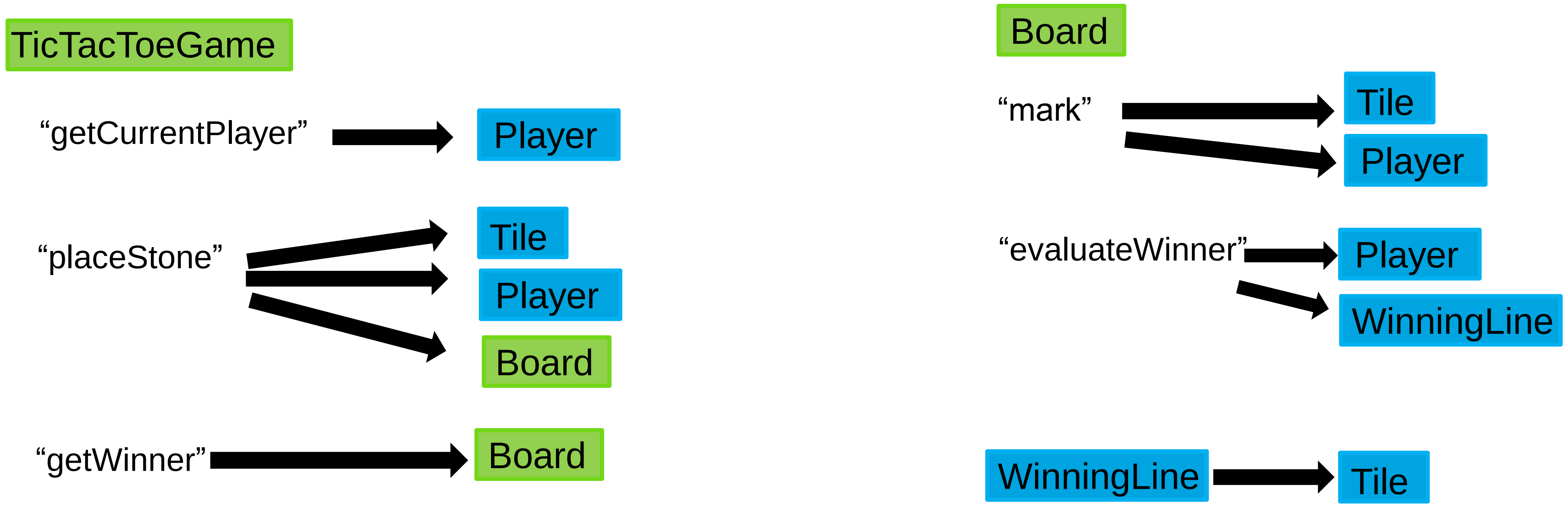
makeXWinTheGameWithThreeXInTopRow
makeOWinTheGameWithThreeOInTopRow
makeOWinTheGameWithThreeOInMiddleRow
haveNoWinnerWhenGameInProgress
makeXWinTheGameWithThreeXInBottomRow
makeXWinTheGameWithThreeXInLeftColumn

Command-Query-Responsibility-Segregation ✓

behavior ?
- relevance
- priority ✓

Separation of Concerns Principle

Identifying orchestrators and actuators
using Object Calisthenics rules like "wrap all primitives and strings in classes"
"wrap collections in classes"
"classes must have state"
"no getter properties" …



Learning

Test Driven Development

Transformation Priority Premise

Object Calisthenics

Find orchestrators and actuators

Command-Query-Responsibility-Segregation

Wrap primitive types and collections in classes

Refactor “if condition” to “loop”