Bowling

The game consists of 10 frames as shown above. In each frame the player has two opportunities to knock down 10 pins. The score for the frame is the total number of pins knocked down, plus bonuses for strikes and spares. A spare is when the player knocks down all 10 pins in two tries. The bonus for that frame is the number of pins knocked down by the next roll. So in frame 3 above, the score is 10 (the total number knocked down)plus a bonus of 5 (the number of pins knocked down on the next roll.) A strike is when the player knocks down all 10 pins on his first try. The bonus for that frame is the value of the next two balls rolled. In the tenth frame a player who rolls a spare or strike is allowed to roll the extra

balls to complete the frame. However no more than three balls can be rolled in tenth frame.

Write a class named “Game” that has two methods

- roll(pins : int) is called each time the player rolls a ball. The argument is the number of pins knocked down.
- score() : int is called only at the very end of the game. It returns the total score for that game.

Rating: Easy

My Solution:

Source: http://butunclebob.com/ArticleS.UncleBob.TheBowlingGameKata

CalcStats

Your task is to process a sequence of integer numbers to determine the following statistics:

- minimum value
- maximum value
- number of elements in the sequence
- average value

For example: [6, 9, 15, -2, 92, 11]

- minimum value = -2
- maximum value = 92
- number of elements in the sequence = 6
- average value = 21.833333

Rating: Easy

My Solution:

Source: http://www.blog.btbw.pl/java/code-kata-10-java-calc-stats/

Even or Odd

Given an array of numbers, determine whether the sum of all of the numbers is odd or even.

Give your answer in string format as 'odd' or 'even'. If the input array is empty consider it as: [0] (array with a zero).

Example:

oddOrEven([0]) returns "even"

oddOrEven([2, 5, 34, 6]) returns "odd"

oddOrEven([0, -1, -5]) returns "even"

Rating: Easy

My Solution:

Source: https://www.codewars.com/kata/odd-or-even

FizzBuzz

Imagine the scene. You are eleven years old, and in the five minutes before the end of the lesson, your math teacher decides he should make his class more “fun” by introducing a “game”. He explains that he is going to point at each pupil in turn and ask them to say the next number in sequence, starting from one. The “fun” part is that if the number is divisible by three, you instead say “Fizz” and if it is divisible by five you say “Buzz”. So now your math teacher is pointing at all of your classmates in turn, and they happily shout “one!”, “two!”, “Fizz!”, “four!”, “Buzz!”… until he very deliberately points at you, fixing you with a steely gaze… time stands still, your mouth dries up, your palms become sweatier and sweatier until you finally manage to croak “Fizz!”. Doom is avoided, and the pointing finger moves on.

So of course in order to avoid embarrassment in front of your whole class, you have to get the full list printed out so you know what to say. Your class has about 33 pupils and he might go round three times before the bell rings for break time. Next math lesson is on Thursday. Get coding!

Write a program that prints the numbers from 1 to 100. But for multiples of three print “Fizz” instead of the number and for the multiples of five print “Buzz”. For numbers which are multiples of both three and five print “FizzBuzz “.

Sample output:

1

2

Fizz

4

Buzz

Fizz

7

8

Fizz

Buzz

11

Fizz

13

14

FizzBuzz

16

17

Fizz

19

Buzz

... etc up to 100

Rating: Easy

My Solution: https://github.com/WolfgangOfner/TDD-Kata/tree/master/FizzBuzz

Source: http://codingdojo.org/kata/FizzBuzz/

Leap year

Write a function that returns true or false depending on whether its input integer is a leap year or not.

A leap year is divisible by 4 but is not otherwise divisible by 100 unless it is also divisible by 400.

Rating: Easy

My Solution: https://github.com/WolfgangOfner/TDD-Kata/tree/master/LeapYear

Source: https://gist.github.com/alastairs/1142957

List comparator

You have two arrays in this kata, every array contains only unique elements. Your task is to calculate the number of elements in the first array which are also in the second array.

Rating: Easy

My Solution: https://github.com/WolfgangOfner/TDD-Kata/tree/master/ListComparator

Source: https://www.codewars.com/kata/array-comparator

Prime Factor

Write a class named “PrimeFactors” that has one static method: generate.

The generate method takes an integer argument and returns a List<int>. That list contains the prime factors in numerical sequence.

For example:

- 100 should return 2, 2, 5, 5
- 2 should return 2

- Smaller than 2 should return an empty list

Rating: Easy

My Solution: https://github.com/WolfgangOfner/TDD-Kata/tree/master/PrimeFactors

Source: http://butunclebob.com/ArticleS.UncleBob.ThePrimeFactorsKata

Queue

Implement your own queue (existing .Net implementations are not allowed (list, array, …)).

The queue should have the following methods:

Enqueue(object): Adds an element to the queue.

Dequeue(): Returns the first element and removes it from the queue.

Peek(): Returns the first element without removing it.

Rating: Easy - Medium

My Solution: https://github.com/WolfgangOfner/TDD-Kata/tree/master/Queque

Recently used list

Develop a recently-used-list class to hold strings uniquely in Last-In-First-Out order.

- The most recently added item is first, the least recently added item is last.
- Items can be looked up by index, which counts from zero.
- Items in the list are unique, so duplicate insertions are moved rather than added.
- A recently-used-list is initially empty.

Rating: Easy - Medium

My Solution: https://github.com/WolfgangOfner/TDD-Kata/tree/master/RecentlyUsedList

Source: https://tddkatas.codeplex.com/documentation

Stack

Implement your own stack (existing .Net implementations are not allowed (list, array, …)

The stack should have the following methods:

- Push(object): Adds an element to the stack.
- Pop(): Returns the last element and removes it from the stack.
- Peek(): Returns the last element without removing it.

Rating: Easy - Medium

My Solution: https://github.com/WolfgangOfner/TDD-Kata/tree/master/Stack

String calculator

Create a simple String calculator with a method **int Add(string numbers)**

1. The method can take 0, 1 or 2 numbers, and will return their sum (for an empty string it will return 0) for example **" " or "1" or "1,2"**
2. Start with the simplest test case of an empty string and move to 1 and two numbers
3. Remember to solve things as simply as possible so that you force yourself to write tests you did not think about
4. Remember to refactor after each passing test
5. Allow the Add method to handle an unknown amount of numbers
6. Allow the Add method to handle new lines between numbers (instead of commas).
    1. the following input is ok: “1\n2,3” (will equal 6)
    2. the following input is NOT ok: “1,\n” (not need to prove it - just clarifying)

Rating: Easy

My Solution: https://github.com/WolfgangOfner/TDD-Kata/tree/master/StringCalculator

Source: http://osherove.com/tdd-kata-1/

Tennis

This kata is about implementing a simple tennis game. I came up with it while thinking about Wii tennis, where they have simplified tennis, so each set is one game.The scoring system is rather simple:

1. Each player can have either of these points in one game 0 15 30 40
2. If you have 40 and you win the ball you win the game, however there are special rules.
3. If both have 40 the players are deuce.
    1. If the game is in deuce, the winner of a ball will have advantage and game ball.
    2. If the player with advantage wins the ball he wins the game
    3. If the player without advantage wins they are back at deuce.

1. A game is won by the first player to have won at least four points in total and at least two points more than the opponent.
2. The running score of each game is described in a manner peculiar to tennis: scores from zero to three points are described as “love”, “fifteen”, “thirty”, and “forty” respectively.
3. If at least three points have been scored by each player, and the scores are equal, the score is “deuce”.
4. If at least three points have been scored by each side and a player has one more point than his opponent, the score of the game is “advantage” for the player in the lead.

Rating: Medium

My Solution: https://github.com/WolfgangOfner/TDD-Kata/tree/master/Tennis

Source: http://codingdojo.org/kata/Tennis/

Tree

Implement a binary tree search algorithm.

Rating: Easy - Medium

My Solution: https://github.com/WolfgangOfner/TDD-Kata/tree/master/Tree

Word wrapper

You write a class called Wrapper, that has a single static function named wrap that takes two arguments, a string, and a column number. The function returns the string, but with line breaks inserted at just the right places to make sure that no line is longer than the column number. You try to break lines at word boundaries.

Like a word processor, break the line by replacing the last space in a line with a newline.

Rating: Easy

My Solution: https://github.com/WolfgangOfner/TDD-Kata/tree/master/WordWrapper

Source: http://codingdojo.org/kata/WordWrap/

Human readable time

Write a function, which takes a non-negative integer (seconds) as input and returns the time in a human-readable format (HH:MM:SS)

- HH = hours, padded to 2 digits, range: 00 - 99
- MM = minutes, padded to 2 digits, range: 00 - 59
- SS = seconds, padded to 2 digits, range: 00 - 59

The maximum time never exceeds 359999 (99:59:59).

You can find some examples in the test fixtures.

- Assert.AreEqual(TimeFormat.GetReadableTime(0),"00:00:00");
- Assert.AreEqual(TimeFormat.GetReadableTime(5),"00:00:05");
- Assert.AreEqual(TimeFormat.GetReadableTime(60),"00:01:00");
- Assert.AreEqual(TimeFormat.GetReadableTime(86399),"23:59:59");
- Assert.AreEqual(TimeFormat.GetReadableTime(359999),"99:59:59");

Rating: Easy

My Solution: https://github.com/WolfgangOfner/TDD-Kata/tree/master/HumanReadableTime

Source: https://www.codewars.com/kata/52685f7382004e774f0001f7/train/csharp

Tic-Tac-Toe Checker

If we were to set up a Tic-Tac-Toe game, we would want to know whether the board's current state is solved, wouldn't we? Our goal is to create a function that will check that for us!

Assume that the board comes in the form of a 3x3 array, where the value is 0 if a spot is empty, 1 if it is an X, or 2 if it is an O, like so:

[[0,0,1],

[0,1,2],

[2,1,0]]

We want our function to return -1 if the board is not solved yet, 1 if X won, 2 if O won, or 0 if it's a cat's game (i.e. a draw).

You may assume that the board passed in is valid in the context of a game of Tic-Tac-Toe.

Rating: Easy - Medium

My Solution: https://github.com/WolfgangOfner/TDD-Kata/tree/master/TicTacToeChecker

Source: https://www.codewars.com/kata/tic-tac-toe-checker

Hashi solver

Write a program that evaluates a provided solution for the game Hashi. The program must check the following conditions:

- Every island has the needed amount of bridges.
- Every bridge is connected to two islands.
- No bridge is crossing with another one.

A description of the game and the rules can be found on Wikipedia.

Rating: Hard

My Solution: https://github.com/WolfgangOfner/TDD-Kata/tree/master/Hashi