# Exercises: Test-Driven Development

This document defines the **in-class exercises** assignments for the ["High-Quality Code" course @ Software University](https://softuni.bg/courses/high-quality-code).

## Custom Data Structure: Research

Implement a popular data structure by your choice. A good starting point is the **Tree** data structure.

You may prefer to implement a relatively simple data structure because the focus of this exercise is not the implementation itself, but the approach to it.

Research on the Internet (for example, in Wikipedia) how this data structure works and what operations it should support (its **public interface**).

## Custom Data Structure: Specification and Tests

After you know what the public interface of your data structure is, write tests which specify how your (not yet written) code should behave.

**Example:** A tree consists of **tree nodes**. Each tree node has **value** and **children**. The **TreeNode<T>** tests may cover the following cases:

* Creating an empty tree node: should have default value and no children
* Creating a tree node with a given value: should store that value and have no children
* Creating a tree node with children
* Creating a tree node, then adding a single child to it: should add the child
* Creating a tree node, then adding more children to it: should add all children

The **Tree<T>** class consists of a single **TreeNode<T>** which is the root. All other items in the tree are children of the root. Write tests for the **Tree<T>** class too.

## Custom Data Structure: Implementation

Your tests will fail initially (or even fail to compile). Write the code for your data structure of choice so that all tests pass. You can write new tests after you've started implementing your data structure. You can also change old tests but prefer not doing it: changing tests means changing requirements and you should know the requirements before you've started writing the code.