

# ROHDE & SCHWARZ TECHNICAL PROFICIENCY (WPF)

Assume the following view should be constructed:

The interface shows a tree view with the following structure:

- ☒ Test 1
  - ☒ Test 1.1
  - ☒ Test 1.2
  - ☒ Test 1.3
  - ☒ Test 1.4
- ☐ Test 2
  - ☐ Test 2.1
  - ☐ Test 2.2
  - ☐ Test 2.3

Buttons: Collapse, Expand, Back, Start

## Desired Functionality

1. The tree should get populated with a given list of tests.
2. The test item display names should be shown next to a check box.
3. The user should be able to check/uncheck the check box.
4. Checking a parent item should check all child items, unchecking a parent should uncheck all children. An indeterminate check state should be supported for only parent items.
5. "Collapse" should collapse the tree to display only the root test items, "Expand" should expand all test items.
6. "Back" should only be enabled if any checkbox is checked, and reset all check boxes.
7. "Start" should only be enabled if any checkbox is checked, and show a popup with the display names for all checked tests.

## Prerequisites

You can assume that an existing flat `IList<Test>` of model objects to display already exists:

```
class Test
{
    int Major;
    int Minor;
}
```

For example, the “Test 1.2” represents `Test { Major = 1, Minor = 2 }`. There exists no `Test` with `Minor = 0`, so there are seven tests represented in above tree: (1,1), (1,2), (1,3), (1,4), (2,1), (2,2), and (2,3).

You can use standard WPF `Button`, `CheckBox` and `TreeView` controls. The necessary icons `collapse.png`, `expand.png`, `back.png`, `start.png`, and `cancel.png` also already exist.

## Exercise

Implement above task compliant with the MVVM pattern using XAML and C#.

In addition to ensuring the proper functionality, the code should be structured in a way allowing for easy unit testing and extension.

You can disregard any styling like colors and font sizes, but the resulting view should have the general structure and alignment.