

Permanente Evaluatie 4 – Questionnaire

Opleiding: Bachelor Elektronica-ICT

Opleidingsfase: 1

Vak: Object Oriented Programming 1 Docent: Ronny Mees

Versie A



In this project you will create a Multiple Choice Questionnaire app in a step-by-step fashion.

Multiple choice questions have been a proven tool in a teachers toolbox for unexpected tests, exams, quizzes and so on. While it does take some effort to create good questions and even more time to create good possible answers, they have proven to be less time-consuming when it comes to correcting these tests. On top of that they are perfect for digital testing of student knowledge.

So, let's prepare you for the exam of Object Oriented Programming 1 by creating an application that can train you with multiple choice questions.

A multiple-choice question usually includes a statement or question followed by 4 or 5 choices. You must select the best answer from the choices given.



Source: <https://www.educationcorner.com/multiple-choice-tests.html>

The goal of this challenge is to create a couple of classes as part of a library, which can be used to create a graphical application. These classes and graphical application will be described step-by-step in this challenge.

A CONSOLE APPLICATION

Start by creating a new **Console Application** within a solution and add projects from there on.

This application serves as a demonstrator of the **Questionnaire** library and should provide some code snippets that show how to use the library.

Make sure the solution and the application have a decent name.

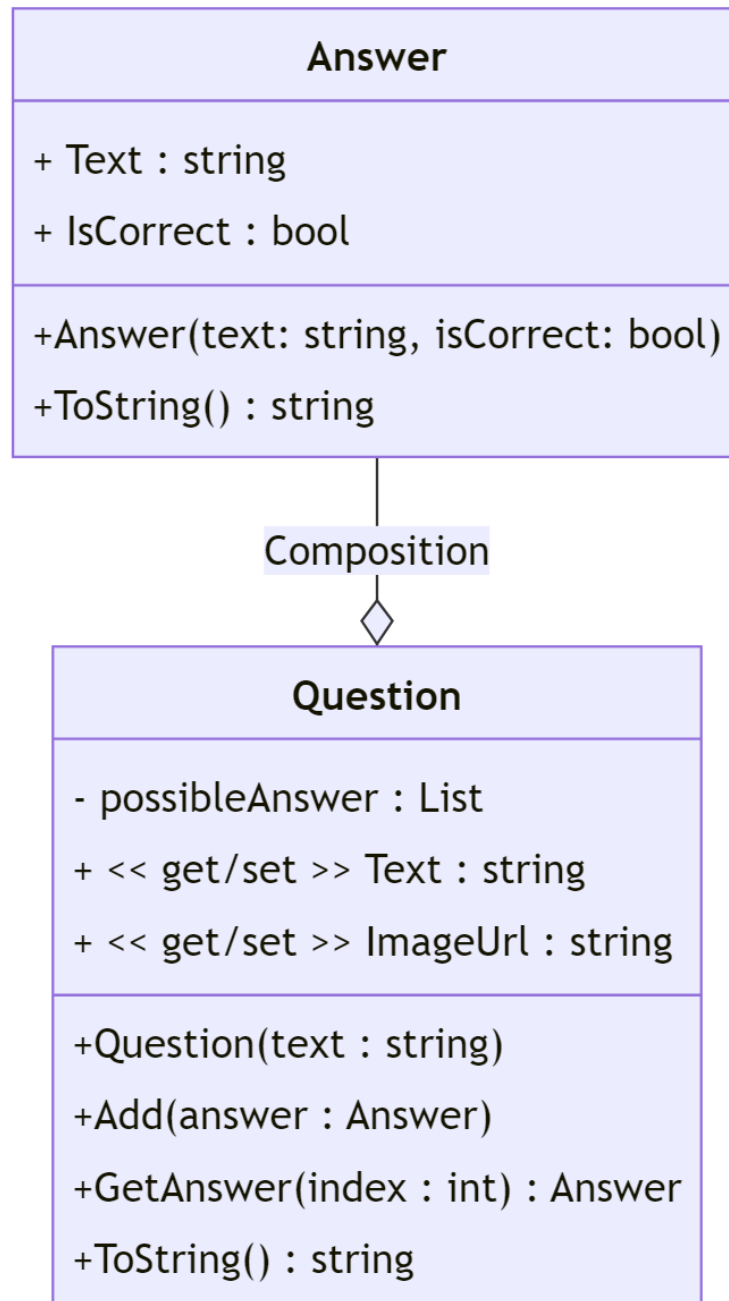
A QUESTIONNAIRE LIBRARY

Create a library as described in [Chapter 42 - Creating Libraries](#).

QUESTION AND ANSWER CLASSES

Basically a multiple choice questions consists of an actual question and a number of possible answers (typically 4 but we should not depend on that).

So let's take a look on how we can model this using UML so we can implement these classes.



Note that this is the bare minimum UML diagram of both these classes. Feel free to extend these classes as you see fit. It is your job to make sure they are usable by applying the knowledge of your OOP1 course and the best-practices that have been taught to you.

QUESTIONNAIRE - THE GAME

Create a graphical WPF application that allows the user to solve some pre-programmed multiple choice questions.

Questionnaire – A student knowledge testing tool

Read the question and select the answer that you think is correct.


Who played Jack O'Neill in Stargate ?

Brendan Fraser

Richard Dean Anderson

Kurt Russell

Brendan Fraser



2/12

Application developed by Chuck Norris – Student @ VIVES University College

Do not try to copy this exact interface. Make it your own ! You can add more information, images, controls, ...

Some things to keep in mind:

- Create your own style but keep it user-friendly and clean
- Provide an About window that shows some basic information about the application (author, year, description, link to github, ...). Allow the user to access this window.

TRIVIA API

To get some random questions we can use any API that provides multiple choice questions. A good example is <https://the-trivia-api.com/>.

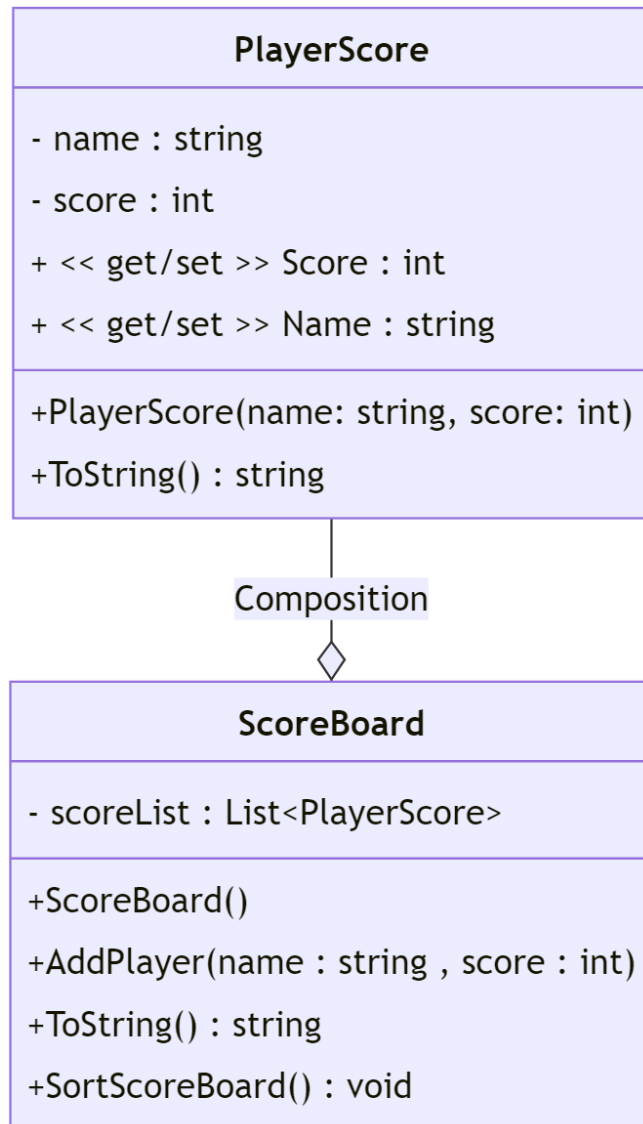
Since making REST requests in C# is not a beginners topic, a library was created for those who wish to use it. It is included with this assignment, copy the **TriviaApiLibrary** directory in your solution. Make sure to import it into your solution and to set it as a dependency to your WPF app.

To fetch a question from the API you need to:

- Call the static `RequestRandomQuestion()` method of `TriviaApiRequester` from for example a `Button` click event handler
- Implement the `IQuestionHandler` interface in your WPF Window to process the retrieved question

SCOREBOARD

Before you can implement a scoreboard you will need to add a `Scoreboard Library` with a `Scoreboard` class similar to this UML diagram:



- `AddPlayer()` : adds a new entry to the scoreboard.
- `SortScoreBoard()` : sort's all entries on score in descending order

When a player has answered for example 10 questions, show a small scoreboard Window with his/her stats.

You can create fake scores for the other entries or you can try to save and load the scores from a file for extra grades.

THE README

Last but not least you will also need to provide a **README.md** file in your repository with the following sections:

- Project description: a short description of the whole solution and the different projects in the solution
- Author: Your name
- Screenshots: a couple of screenshots of the application
- Setup and Usage: short description of what applications and tools are required to use the app (Visual Studio, ...)
- UML Class diagrams of the classes in the library
- Future Improvements: explain the things that do not work or what could be done better