University of Southern Denmark
The engineering programmes in Sønderborg

Advanced Object-Oriented Programming

Spring EXAMINATION

Tuesday, 10 June 2025

Problem 1 starts on the next page!

General Information

In general, for all tasks you can only use C# and the following libraries:

The .Net System library, Avalonia and CommunityToolkit.MVVM.

Problem 1 – Object-Oriented Analysis (20 Points)

In the folder `Problem_1_OOP\DocumentManager\` you can find a fully functional C# console project called DocumentManager. Analyze the code to answer the following questions. Write your answers in the provided text file `Problem_1_Submission.txt` in the Problem1 folder.

1.1. General Analysis (2 Points)

In one or two sentences, briefly describe in your own words what real-world concept or system this code models.

1.2. Object Oriented Programming (2 Points)

Look at the interfaces defined in `Problem_1_OOP\DocumentManager\Interfaces.cs` and their implementations in the rest of the code. Briefly describe their purpose in this application.

1.3. OOP Principles (8 Points)

For each of the 4 basic OOP principles:

- State whether it is present in any way in the code provided.
  (4 points / 1 point Each)
- If present, briefly describe how and where (give at least one example from the code) and what purpose it serves in this specific code.
  (4 points / 1 point each)

1.4. SOLID Principles (4 Points)

Identify two different SOLID principles that are demonstrably applied in the code example provided. For each one: Name the Principle and explain its purpose and how it specifically benefits this example.

1.5. Design Patterns (4 Points)

Identify **one** design pattern that is implemented in the provided code example. Name it and describe its role within this specific code example.

Problem 2 – UI (30 Points)

In the folder `Problem_2_UI\RectangleUI\` you will find a basic Avalonia project *RectangleUI* that has a finished user interface, but no actual functionality.

The goal of this application is to allow a user to display an arbitrary number of rectangles at random positions. The rectangles' colors and positions will change periodically, and they will move randomly.

Follow the subtasks below and implement the functionality.

!!! IMPORTANT !!!

Strictly follow the MVVM pattern, which means you should only modify the following four files.

- `Problem_2_UI/RectangleUI/ViewModels/MainWindowViewModel.cs`
- `Problem_2_UI/RectangleUI/Models/RectangleData.cs`
- `Problem_2_UI/RectangleUI/Views/MainWindow.axaml`
- `Problem_2_UI/Problem_2_Submission.txt`

Any code you add or modify in other files will not count toward your solution. For example, do not modify the CodeBehind (`MainWindow.axaml.cs`).

2.1. MVVM Pattern (5 Points)

In a few sentences, explain what the MVVM pattern is used for and what the Model, View, and View Model do in this app. Use your own words, and write them in the file named `Problem_2_Submission.txt.`

2.2. Basic Functionality (15 Points)

Modify the project so that each time a user clicks the Add button, a new rectangle appears on the canvas. Follow the requirements below:

- The original layout of the application should not change.
- Each rectangle should appear at a random position with a width and height according to the currently selected values on the given sliders. There is one slider for the Width and one for the Height of the rectangles.
- Each rectangle should always be completely visible. This means that the rectangles cannot be shown outside the canvas boundaries.

Hint: Everything in the MainWindow.axaml is basically already set up. You just need to modify some parts to add the correct bindings.

2.3. Advanced Functionality (10 Points)

Modify the project so that the rectangles change position and color every 2 seconds. Follow these requirements:

- The original layout can stay the way it is and everything from Subtask 2 should continue to work.
- Make sure that the UI is responsive and that changes to the UI happen only on the UI thread.
- The colors can simply come from a list of predefined colors and do not have to be random RGB values.
- Even if you change the position of the rectangles, make sure that each rectangle is always completely visible. This means that the rectangles cannot be displayed outside the canvas boundaries.

Problem 3 – Unit Testing (15 Points)

In the Folder `/ Problem_3_UnitTesting/Counter/` you will find an Avalonia application called Counter. It has a TextBlock that shows a number and two buttons: one to increase the number by one and one to decrease the number by one. The Decrement button only works if the number is higher than 0. This is so the counter cannot be negative.

3.1. Basic Unit Testing (10 Points)

Create a new XUnit Testing Project and make sure it correctly references the Counter Project.

Then Create a *"normal"* (not Avalonia) XUnit Test Class for the `MainWindowViewModel` and write the following 5 tests, that make sure that:

- The counter is correctly initalized to zero
- The `DecrementCommand` cannot be executed after initialization
- The `IncrementCommend` correctly increments the counter by one
- After Incrementing the `DecrementCommand` can be executed
- The `DecrementCommand` correctly decrements the counter by one

3.2. Avalonia Unit Testing (5 Points)

Setup Headless Avalonia for XUnit testing and write an Avalonia UI test that simulates a user clicking the increment button 100 times and then check whether the output displays the expected result. Make sure that the button and the output actually exist.

Problem 4 – JSON and LINQ (35 Points)

You are given a JSON file (`Problem_4_LINQ/data/spaceship.json`) that contains a database on spaceships and their trips through the solar system.

4.1. Parse JSON (10 Points)

Create a new C# Console Application and parse the JSON into a list.
Be aware that there might be missing values.

4.2. Run Queries (20 Points)

Create, run, and print the results of five LINQ queries on your list. The queries should return the following:

1. All military ships. **(2 points)**
2. All spaceships that are currently traveling *(note: these spaceships do not have an arrival date yet)*. **(4 points)**
3. All spaceships sorted by the number of trips they have taken. The spaceships with the most trips are at the top of the list. **(4 points)**
4. The average number of trips taken for each type of ship. **(5 points)**
5. All ships that departed Ganymede Port in the year 2245 **(5 points)**

4.3. Binary Search (5 Points)

Run a binary search for the ship with the name "Rocinante" and print the resulting ship if found.