Practical Task 5.4

(High Distinction Task)

Submission deadline: 10:00am Monday, May 18
Discussion deadline: 10:00am Saturday, May 30

General Instructions

This practical task extends Task 5.3. After showing the basic Reaction-Timer machine to potential customers, the marketing department has found that some of them would like to pay more for a deluxe Reaction-Timer that allows multiple games after payment of a single coin. In this machine, when a coin is inserted, the player is allowed to play three games. The operating sequence is very similar to the previous one:

- insert coin,
- press the Go/Stop button,
- wait random delay,
- press the Go/Stop button, then
- display the time for three seconds.

If the machine has not yet completed three games, it then displays 'Wait...'. Subsequently, it waits a (new) random delay and proceeds further as in the first game. After displaying the time for the last game, the machine shows the average time 'Average = t.tt' (e.g. Average = 0.57) for five seconds, then the game is over. The enhanced controller should fulfil the following requirements.

- If after inserting the coin the player fails to press Go/Stop within ten seconds, the game is over.
- If the player presses the Go/Stop button during the waiting period, the game is aborted, and the average value is not displayed. Once again, no reward for cheating!
- If the player presses the Go/Stop button while the machine is displaying a reaction-time value, the machine immediately moves on to the next game (or shows the average time) without waiting for the full three seconds of display time.
- If the player presses the Go/Stop button while the machine is displaying the average time, the game is immediately over.

1. As in the original task, you are expected to start with the design of the event-driven system. Specifically, revise your current state-transition diagram and decide whether the existing states are sufficient to model the work of the enhanced system. Are there any new specific transitions that need to be handled by the controller? When you trust your design, implement it in code. You must produce a new `EnhancedReactionController` class written in the `EnhancedReactionController.cs` source code file.
2. Because this is a high distinction task, it does not provide you with a set of tests as it was done for the basic Reaction-Timer machine. Actually, you are expected to propose and write your own test cases similar to what you have seen previously in Task 5.3. You should not just copy and paste the original tests. Develop your own test instances that you feel important and well-motivated. Thus, we encourage you to thoroughly test your code against both expected and unexpected scenarios. Your work on testing will result in a separate project with `Tester` as a new main class to check the required `EnhancedReactionController` class. Please, make sure that you comment every single test that you add stating its purpose. You must have enough tests: The total number should not be less than what you found in Task 5.3 though you should also focus on the effectiveness of the tests, not just the quantity.

Further Notes

- Explore the attached book chapter entitled "Notes on Finite State Machines" to learn about the concepts you will need to complete this project correctly. Especially, focus on the examples of the state-transition diagrams as they will help you sketch your FSM.
- The following video materials will give you more insights on the finite state machines, their relevance to the state design pattern, and how to implement the pattern in code.
  - https://www.youtube.com/watch?v=N12L5D78MAA
  - https://www.youtube.com/watch?v=MGEx35FjBuo
  - https://www.youtube.com/watch?v=rs7DXnEmHMM
  - https://www.youtube.com/watch?v=1CAO-l6k-j0
- The following links are to cover several nuances that will likely encounter while working on this task.
  - https://www.geeksforgeeks.org/nested-classes-in-c-sharp/
  - https://www.codeproject.com/Tips/875393/Static-Dynamic-Dispatch-ReExplained
  - https://www.tutorialsteacher.com/csharp/csharp-interface
  - https://www.dotnettricks.com/learn/designpatterns/state-design-pattern-c-sharp
  - https://dotnettutorials.net/lesson/state-design-pattern/

Marking Process and Discussion

To get your task completed, you must finish the following steps strictly on time.

- Make sure that your program implements the required functionality. It must compile and have no runtime errors. Programs causing compilation or runtime errors will not be accepted as a solution. You need to test your program thoroughly before submission. Think about potential errors where your program might fail.
- Submit a picture of the state-transition diagram that you prepared as part of designing the enhanced controller.
- Submit the expected code files as an answer to the task via OnTrack submission system. Cloud students must record a short video explaining their design, work, testing, and the resulting solution to the task. Upload the video to one of accessible resources, and refer to it for the purpose of marking. You must provide a working link to the video to your marking tutor in OnTrack.
- On-campus students must meet with their marking tutor to demonstrate and discuss their solution in one of the dedicated practical sessions. Be on time with respect to the specified discussion deadline.
- Answer all additional questions that your tutor may ask you. Questions are likely to cover lecture notes, so attending (or watching) lectures should help you with this **compulsory** interview part. Please, come prepared so that the class time is used efficiently and fairly for all students in it. You should start your interview as soon as possible as if your answers are wrong, you may have to pass another interview, still before the deadline. Use available attempts properly.

Note that we will not accept your solution after the submission deadline and will not discuss it after the discussion deadline. If you fail the deadline, you also fail the task and this may impact your performance and your final grade in the unit. Unless extended for all students, the deadlines are strict to guarantee smooth and on-time work throughout the unit.

Remember that this is your responsibility to keep track of your progress in the unit that includes checking which tasks have been marked as completed in the OnTrack system by your marking tutor, and which are still to be finalised. When marking you at the end of the unit, we will solely rely on the records of the OnTrack system and feedback provided by your tutor about your overall progress and quality of your solutions.