Lehrstuhl für Computergestützte Modellierung und Simulation
TUM School of Engineering and Design
Technical University of Munich

Professional Software Engineering

Lecture 7: Exercises

Exercise 1 Delegates

Use this exercise to familiarize yourself with delegates, before moving onto Exercise 2.

Program a generic delegate with the name *AdditionHandler* which takes in functions with the following signature, where *Type* can be an integer, double etc.

public Type Add(Type firstArg , Type secondArg);

Initialize three instances of *AdditionHandler* and use them to add integers and doubles, as well as concatenate strings. Once you are done, try using the three instances of *AdditionHandler* and print the output to the console.

Hint 1: The addition operator (+) can be used to concatenate two strings.

Hint 2: You can use lambda operators instead of the conventional method structure as shown below.

```
//both methods are equivalent
public static int AddThreeToNumber(int number) {
    return number + 3;
}
public static int AddThreeToNumber(int number) => number + 3;
```

Exercise 2 Events

Note: This exercise builds upon Lecture 1's code-along exercise (traffic lights example).

In this exercise, you will program the event of pressing a pedestrian crossings' button in C#.

Create the following classes in your program.

1. PublisherCrosswalkButton
2. ListenerTrafficLight
3. ListenerCrosswalkLight

ListenerTrafficLight and *ListenerCrosswalkLight* should subscribe to the event programmed in the class *PublisherCrosswalkButton*. Once the event is raised, both listeners should print to the console the state of their own lights (whether it is green or red). This state should depend on the crosswalk button's state in *PublisherCrosswalkButton*.

Hint: Use a boolean to represent whether the crosswalk button is pressed, e.g., when the button is pressed, set the boolean variable to True.