Notes

Exercises about Events and Delegates

- Solve them in Visual Studio.

Exercise 17.01

- Create a first a Console application.
    - Later a Windows Forms or a WPF application.

	Try to do all three. You will see that there are several different solutions possible. *With the console application you need to work with events. The Windows Forms and WPF application is basically already with events imbedded in their core.*

- We will create a clock. On that clock you can set several alarms. When a certain timestamp is passed by for one of the alarms, there must be an event triggered. This event sets of the alarm. Make sure that the event is only triggered once.
- Test your application by creating 2 clocks all having 2 alarms set. Make sure you don't have to wait long before alarm goes off.
    - Use beeps or console message to prove that it works. 😊
- Tip: Create a class "Clock".
- Tip: This class can have a timer, a list of alarms and if needed a Boolean that marks that a certain alarm has been triggered.
- Tip: Create a delegate for the alarm (give this a good name) inside the class Clock.
- Tip: Create custom event arguments for the alarm. Give this also a good name.
- Tip: Use a timer to check every second if there must be alarm going off or not. The namespace System.Timers will help you. On the

timer you can use the Timer.Elapsed event. Every second, do something.

- Tip: Don't forget to set the timer to enabled. Otherwise nothing will happen.

	The design pattern of this exercise is exactly the same as the given example AdvancedEventHandler. - *You need to create a custom EventHandler as delegate.* - *You need to create an event based on that created custom EventHandler. The alarm has to invoke some method (let the alarm go off).*

Variant 1

- Can you give your clocks different alarm sounds or messages?
  - Change the constructor for it.
  - Use maybe an enumeration for the different kind of methods of giving an alarm.
  - Maybe you can set several kinds of alarms at the same time. You show a message on the screen and you give a sound, depending on the value of the enumeration.
- Can you make the "Snooze" functionality?
  - Meaning. The alarm is set, question is asked to set off or to snooze.
  - When snooze is chosen, the alarm goes off after 30 seconds.
  - When set off, the alarm is "killed".

Notes

....

....

....

....

....

....

....

Exercise 17.02

- Create a screen. Windows Forms or WPF application.
- The screen contains 9 buttons.
    - In a 3 by 3 square.
- When you move over a button with the mouse, the button change colour.
- When you are not moving with the mouse over the button, it changes back to its original colour.
- When you first click on a button, the text becomes an “X”.
- When you click again on a button, the text changes into an “O”.
- Click again, and it is an “X” again.
- When you have a 3 in a row, you win the game.
    - This can be 3 “X”, but also 3 “O”.
- Add a button that resets the game to the start position.

Variant 1

- Add the concept player to it.
- Two players are playing against each other, until the last click has a winner.

Variant 2

- Can you create a computer player that always wins?

Variant 3

- What happens if 2 computer players (that are always winning) plays against each other?

Notes

Exercise 17.03

- Create a first a Console application.
  - Later a Windows Forms or a WPF application.

	Try to do all three. You will see that there are several different solutions possible. *With the console application you need to work with events. The Windows Forms and WPF application is basically already with events imbedded in their core.*
	Define a battle plan, the agile way of working. *Start small and let the project grow.*

- You are creating stock places. Every stock place can contain a maximum number of articles in that stock place.
- The maximum amount of stock places is 10.
- The idea is that there is only one article in a stock place, but it can be there a maximum number of times. The article decides what is the maximum amount in the stock place.
- Once a stock place is created, the first add of articles, defines the kind of article that is in that stock place.
- Once a stock place is empty again, another article can be added to it.
- So you need to implement adding and removing a number of articles from the stock place.
- You can add for example 10 articles to an empty stock place, you can add 5 articles to a stock place where there is already that kind of article in it.
- But you never can exceed the maximum number of articles in a stock place and you can't add different articles to the same stock place.

Notes

- You create an event when a stock place reached 90% or more of the maximum capacity. This is shown as a warning message on the screen.
- When a stock place is completely full, another event is triggered.
- This new event, does this, when possible, a new stock place is created for the same article and the rest of the add is placed in that new stock place.
- Create also an event, when the stock is completely full. Meaning, you can't add a certain product anymore.
  - This can be because there are no empty stock places.
  - This can be because there is no room anymore to the stock place where you can add a part of it.

An example (not all scenarios that must be covered)

- You create an article. For example "Jerrycan". There can be 100 items in 1 stock place.
- You add 10 Jerrycans to a stock place that is empty.
- You add 85 Jerrycans to the same stock place.
  - This must result in a warning message.
- You add another 20 Jerrycans to the same stock place.
  - This must result in a second stock place that is started with Jerrycans. The first is full, the second contains 15 Jerrycans.
- You remove 25 Jerrycans from the stock.

	Plan this exercise carefully. You can use a lot of different solution techniques. This is an exercise of events 😊. *You can freestyle a lot, but the concept must work.*

Notes

Exercise 17.04

- Create a first a Console application.
  - Later a Windows Forms or a WPF application.

	Try to do all three. You will see that there are several different solutions possible. *With the console application you need to work with events. The Windows Forms and WPF application is basically already with events imbedded in their core.*
	Define a battle plan, the agile way of working. *Start small, and let the project grow.*

- You are creating phone networks. Every phone network has a unique number existing of 2 digits.
- Every phone network contains phones. A phone has a number of 3 digits, a name connected to it and a kind of ringtone.
  - You don't have to work with ringtones, you can also use console messages 😊.
  - So every phone has its own messages or tones.
- You can call another person on the same network using 3 digits.
- You can call another person on another network with 5 digits.
- This must happen whenever you call a person with a phone:
  - When you call yourself, you get the message that the person is occupied.
  - When you call a non-existent phone number, you get the message that there can't be a connection made.
  - When you call an existing phone number, whatever network, that phone is starting its ringtone or a message. You also see

Notes

....

....

....

....

....

....

....

on the screen that phone number x (with name) is calling phone number y (with name).

- Every message on the console must show on what phone the message appears.
- You can stop the application when the text “stop” was typed.

Variant 1

- When somebody calls, you are asked the question to pick up the phone. Or not.
- When you pick up, at that moment the call starts.
- With a certain keystroke the call is finished. (In Windows Forms or WPF use a button)
  - The caller can finish with an “X”.
    - When an X is pushed and there are more than one calls busy, the phone number of the caller must be typed to stop that specific call.
  - The called can finish with an “Y”.
    - When an Y is pushed and there are more than one calls busy, the phone number of the called must be typed to stop that specific call.
  - When a call is stopped, the caller must see on the screen what the duration of the call was.
    - For every second the caller pays 1 cent if the call was inside the network. If not, the price is 2 cents.
- When the application stops (you typed stop), you see an overview of all the costs that must be paid by whom.

Notes

Exercise 17.05

- Make a class diagram for the classes below.
- You have a class for a game character.
    - The game character can fight with one weapon.
- A Queen is a game character.
- A King is a game character.
- A Troll is a game character.
- A Knight is a game character.
- You have an interface for the behaviour of a weapon.
- A Knife implements the behaviour for a knife.
- A BowAndArrow implements the behaviour for a bow (and arrow).
- An Axe implements the behaviour for an axe.
- A Sword implements the behaviour for a sword.

	Make sure that every game character can set its weapon and the behaviour for it.

Extra info

- A King is an extension of a game character.
- A Sword (behaviour) is an implementation of a weapon (behaviour).
- A game character has a weapon (behaviour).

Notes

Exercise 17.06

	The starting point of this exercise is 00901-i MultipleDelegates. *Here an event triggers first 3 and then 2 delegates (created in an array).* *List the delegates that are currently attached (invokable) with the event.* *So the result is, to give a list of the actual methods that are called when an event is triggerd*

- There are several solutions.
- Notice that this exercise does not use add and remove handlers.
- You can expermiment en rewrite the whole exercise, but the end result must stay the same way.
    - First the event triggers 3 methods.
    - Then the event triggers 2 methods.

Notes