C# OO - zelfevaluatie
==========

Naam: Ruben Van der Kelen
Datum: 11/01/2020
Volgnummer blok: Blok 2

Onderwerp: Trading Card Game

Stand van zaken:

Alle logica zit in mijn programma (alles met klassebibliotheken), GUI kan enkel de objecten al aanmaken en correct weergeven. Het meerlagenmodel heb ik met interfaces volledig toegepast.

Sterke punten:

Klassebibliotheken, Meerlagenmodel, gebruik van interfaces (als type, voor loose coupling), klassehierarchie (abstracte klasse Kaart), Dependency Injection en IOC.

Aandachtspunten in je aanpak:

Soms wat omslachtig, onoverzichtelijk. Mijn dataopslag en access laat veel te wensen over.

Technische aandachtspunten:

Werking van Streams en Generic Collections, Deep Cloning zorgt voor problemen.

Leerdoelen:

< opsomming van de leerdoelen voor dit blok met het niveau waarop je dat doel al verworven hebt:

- Debugging: inzicht (zie mijn bijvoegsel hieronder)
- Enumerations: inzicht
- Architectuur van een toepassing - Meerlagenmodel: inzicht
- Generic collections: toepassing
- Interfaces : inzicht
- 'Value' en 'Reference' types, cloning van objecten: reproductie
- Klassen – klassehiërarchie: inzicht
- Delegates: reproductie
- Lambda expressions: reproductie

>

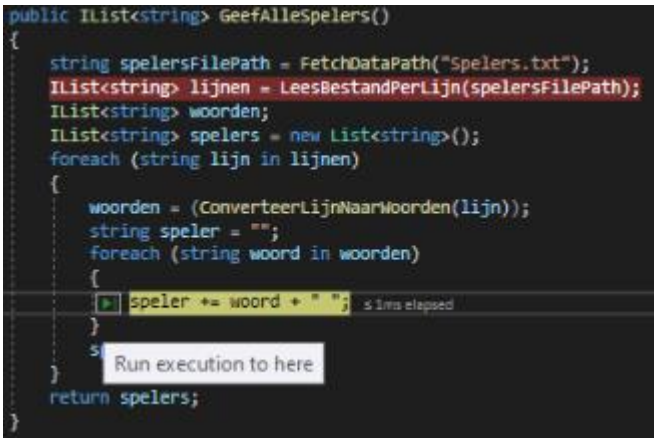
Bijvoegsel Debugging: Hopelijk is dit in orde voor debugging aangezien ik dit niet meer kan komen voordoen?

Debugging C#OO

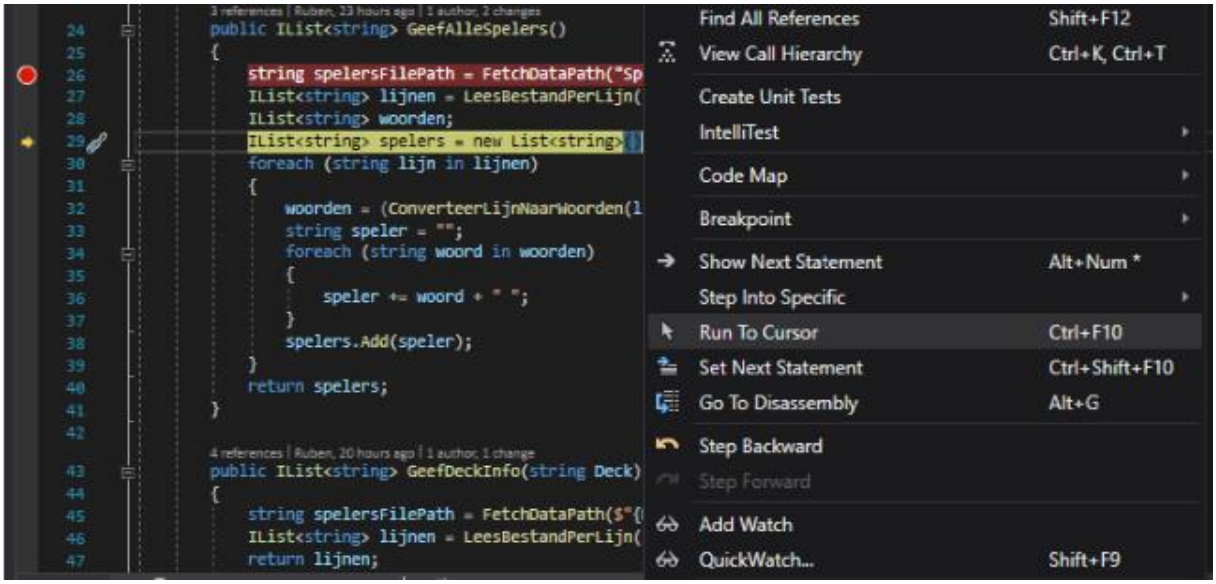
1) Code stap voor stap uitvoeren

1.1) Run To Click / Run To Cursor

Bij Run To Click gaat hij alle code uitvoeren voor de geselecteerde lijn en daarop stoppen, dit kan men doen doorop het groene pijltje te klikken zoals hieronder: (Alle afbeelding afkomstig uit mijn code)



Bij Run To Cursor gebeurt hetzelfde als Run To Click maar door eerst op rechtermuisknop te klikken (naast de code waar gestopt moet worden) en dan Run To Cursor te selecteren, zoals hieronder:



1.2) Step into, Step over, Step out

De volgende mogelijkheden om de code stap voor stap uit te voeren zijn: 1. Step into, 2. Step over, 3. Step out, deze zijn bruikbaar via een taakbalk boven de code tijdens het debuggen, hier staan hun symbolen (in de zelfde volgorde): 

Bij step into (1) gaan we binnen een functie op de huidige lijn gaan en in die context verder debuggen, in mijn voorbeeld zou dit ons naar de LeesBestandPerLijn methode sturen.

Bij step over (2) gaat de functie helemaal uitgevoerd worden en gaan we direct naar de volgende lijn.

We gebruiken step into als we de volledige functionaliteit van de methode (alle stappen die gebeuren) willen nakijken en step over als enkel het resultaat van belang is.

Bij step out (3) gaan we uit de huidige functie stappen en terug gaan naar de functieaanroep.

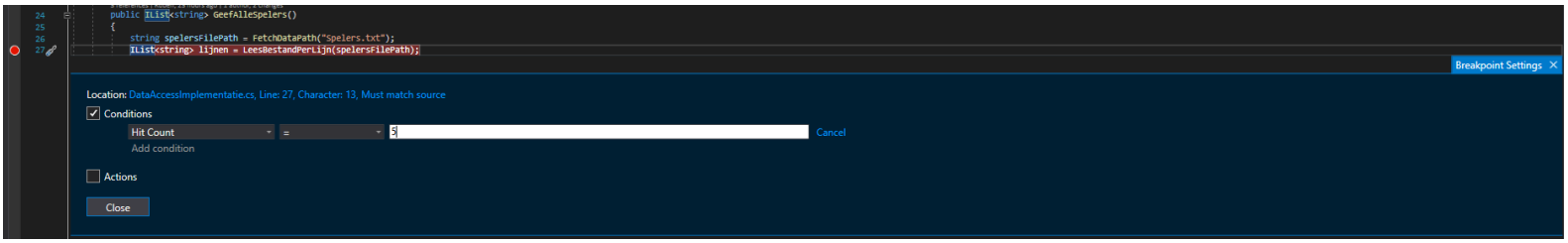
2) Breakpoints

2.1) Vaste Breakpoints

Om een breakpoint te plaatsen doen we -> rechtermuisklik -> Breakpoint -> Insert Breakpoint (dit is dan een vaste breakpoint) Breakpoints plaatsen we best waar we denken dat er iets foutloopt: dit is meestal bij het instellen van parameters (waarden) of bewerkingen met waarden. Deze waarden kan men weergeven door het Locals venster (zie punt 3) De waarde van variabelen bekijken tijdens de uitvoering van je programma).

2.2) Conditionele Breakpoints

Om van onze vaste breakpoint een conditionele breakpoint te maken doen we rechtermuisklik op breakpoint -> conditions, waarna we de condities kunnen instellen zoals hieronder:

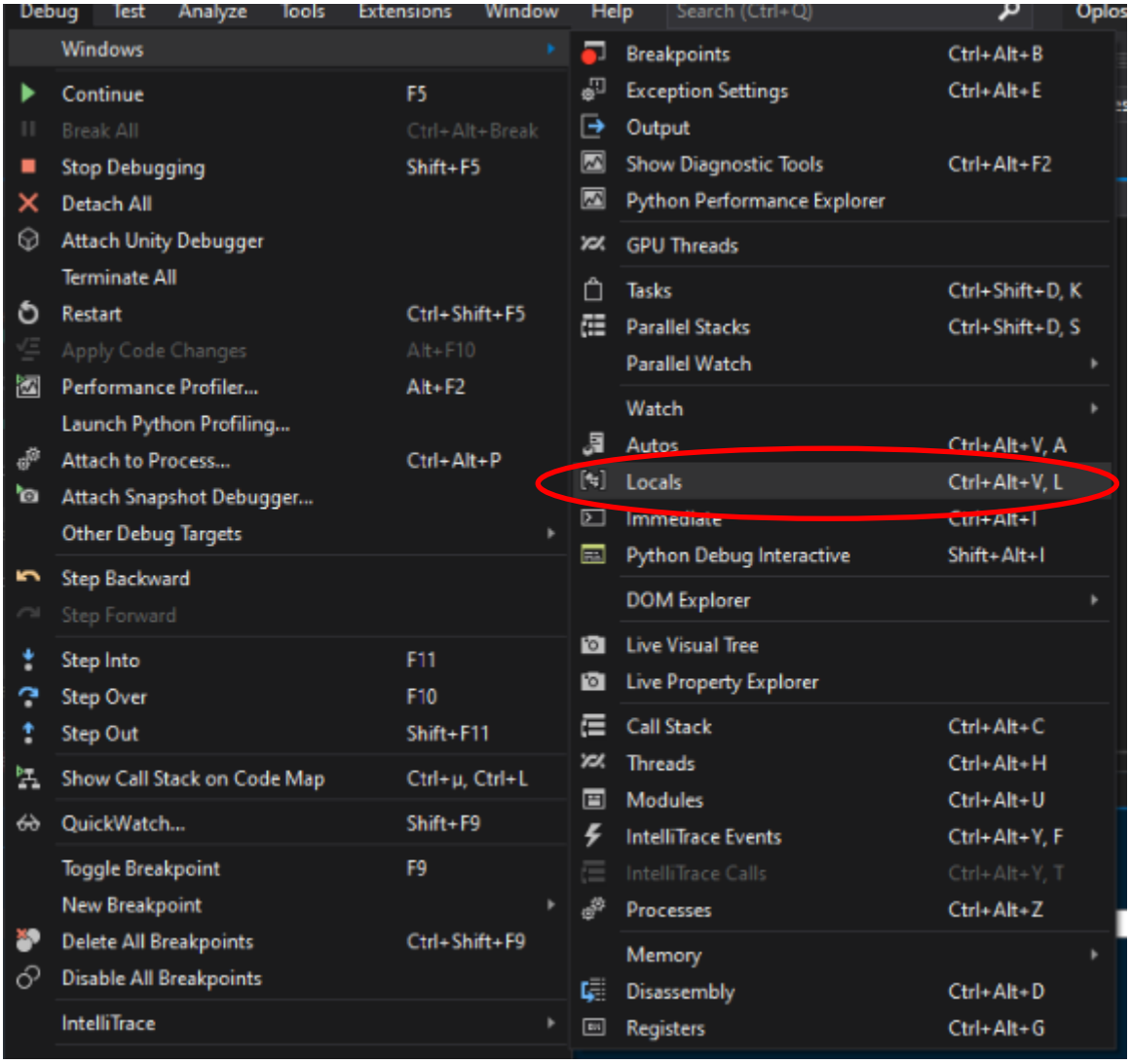


Zo kan men bepalen wanneer de breakpoint geactiveerd wordt: Hit = nadat die x aantal keer gepasseerd is, Conditional Expression = wanneer er aan een bepaalde conditie voldaan is, Filter = wanneer een bepaalde variable gelijk is aan een gekozen waarde.

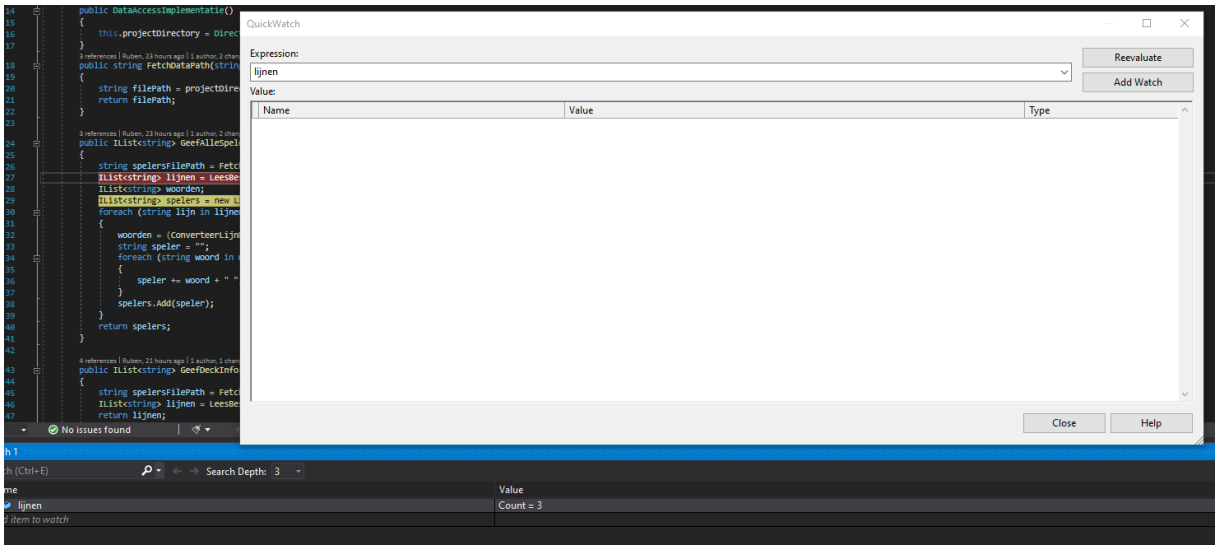
3) De waarde van variabelen bekijken tijdens de uitvoering van je programma

Om de waarden van variabelen te bekijken tijdens het debuggen gebruiken we het locals venster dit kunnen we op twee manieren open:

1 (Taakbalk boven) Debug -> Window -> Locals



2) rechtermuisklik ergens in code -> Quickwatch -> hier kan men locals toevoegen zoals voorbeeld hieronder:



Nog een voorbeeld van locals venster: