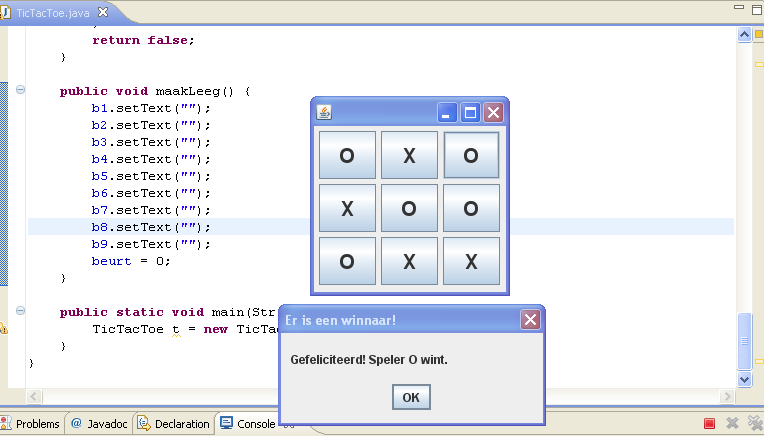
# Practicumbundel .NET Programmeren 2011-2012

Auteur: Michiel Borkent (correcties en suggesties naar: [michiel.borkent@hu.nl](mailto:michiel.borkent@hu.nl))

Laatste wijziging: 30-5-2013

# Week 1

Bestudeer het bestand TicTacToe.java (te vinden op SharePoint) en probeer het uit in Eclipse (of met een ander programma waarmee je deze Java class kan compileren en uitvoeren). Het betreft hier het spel “Drie op een Rij” of in het Engels: “Tic Tac Toe”. Hieronder zie je een afbeelding van het spel en een melding dat speler O gewonnen heeft.



De Java-code is een uitwerking van een practicum Java uit het eerste jaar. In dit practicum gaan we dit spel nabouwen in Visual Studio met de programmeertaal C#. De eerste opdracht is vrij rechttoe rechtaan. Je kunt de code uit Java bijna 1-op-1 overnemen, met uitzondering van de grafische user interface. Deze wordt gebouwd met de Designer van Visual Studio als Windows Forms-applicatie.

In opdracht 2 gaan we het programma anders opzetten. De logica verschuiven we naar een library zodat deze onafhankelijk wordt van de presentatie. Daarna gaan we het Windows Forms-gedeelte aanpassen, zodat alleen nog gebruik maakt van de library. Ook gaan we een simpele Console-versie van “Tic Tac Toe” maken, gebruikmakend van dezelfde library.

## Opdracht 1

1. Maak een project (en eventueel een nieuwe solution aan als je er nog geen hebt) en kies de optie “Windows Forms Application”.
2. In het project staat een C# (.cs) bestand met de naam Form1.cs en ook een bestand met de naam Form1.Designer.cs. Dat laatste bestand bevat een gedeelte van de class Form1. Dit gedeelte wordt gegenereerd wordt door de Designer van Visual Studio. Er staat o.a.: partial class Form1.

Datzelfde zie je ook staan in het bestand Form1.cs. In C# is het mogelijk om code van één bepaalde class te verdelen over verschillende files door middel van het keyword partial.

De gebruiker kan in het bestand Form1.cs zijn eigen code toevoegen terwijl de Designer de gegenereerde code in Form1.Designer.cs opslaat. Op deze manier kan gegenereerde code en gebruikerscode van één en dezelfde class worden gescheiden in verschillende bestanden en blijft het een overzichtelijk geheel.

De file Form1.cs kun je in Code-modus of Designer-modus bekijken. Klik rechts met de muis op de Form1-window in de Designer en selecteer ‘View Code’ of toets F7 in. Je kunt terug naar Designer-modus door shift-F7 in te toetsen.

1. Voeg in Designer-modus via de Toolbox (links) een Button toe aan Form1. Er wordt een knop gecreëerd met de naam button1. Bekijk de Properties van button1 en leeg de Text property.
2. Je kunt de rest van de negen knoppen creëeren door deze knop te kopiëren en acht keer te plakken. Het is aan te raden om alvorens dit te doen de knop alvast vierkant te maken, zodat alle gekopiëerde knoppen ook vierkant worden.
3. Dubbelklik op button1. Er wordt een methode aangemaakt genaamd button1\_Click. Deze methode zal worden uitgevoerd zodra button1 wordt ingedrukt. Omdat we deze methode willen gebruiken voor alle knoppen en niet alleen voor button1 hernoemen we deze methode naar button\_Click. Let erop dat je Visual Studio alle referenties naar deze methode laat updaten.
4. Voeg in de methode de regel System.Windows.Forms.MessageBox.Show("Er is geklikt!"); toe. Run het programma en test of er een messagebox verschijnt bij het drukken op button1.
5. Bekijk de file Form1.Designer.cs. In de methode InitializeComponent staat de code voor button1 t/m button9. Bij het gedeelte van button1 zie je de volgende regel staan:

this.button1.Click += new System.EventHandler(this.button\_Click);

Deze regel zorgt ervoor dat de methode button\_Click wordt aangeroepen bij het klikken op button1. Met de operator += kunnen eventueel meerdere EventHandlers geregistreerd worden. Meer informatie over EventHandlers is te vinden in het boek op blz 171 en verder.

Zorg ervoor dat de methode button\_Click ook wordt uitgevoerd bij het drukken op de andere knoppen. Controleer of het werkt.

1. De basis is gelegd. De rest van het programma TicTacToe kan nu worden aangevuld in Form1.cs (in Code-modus).

## Opdracht 2

Zoals eerder vermeld gaan we in deze opdracht de logica van TicTacToe loskoppelen van de presentatie. Daartoe moet er een library gemaakt worden die deze logica gaat bevatten met een class genaamd TicTacToeEngine, die je in een GUI- of Console-applicatie kan gebruiken. De class moet voldoen aan het volgende UML-class diagram:

|  |
| --- |
| TicTacToeEngine |
| + Property Status : GameStatus (private setter, public getter)  + “representatie van het bord” (zelf te verzinnen hoe je dat oplost) |
| + Board() : String  + ChooseCell(int) : Boolean  + Reset() : void |

GameStatus dien je als volgt te definiëren:

public enum GameStatus { PlayerOPlays, PlayerXPlays, Equal, PlayerOWins, PlayerXWins }

Toelichting bij het UML-diagram:

De methode Board levert een String-representatie van het bord die in de Console-app gebruikt kan worden.

Met ChooseCell kan een client aangeven welke cel de huidige speler kiest.

Het spel kan gereset worden met de methode Reset.

Een voorbeeld van het verloop van het begin van een spel:

1. Er wordt een object t van TicTacToeEngine aangemaakt
2. Status heeft de waarde PlayerOPlays (we kiezen voor het gemak even dat speler O altijd begint)
3. Speler O kiest cel 1, de client voert bool result = t.ChooseCell(1) uit
4. Aan de hand van de bool result kan worden bepaald of speler O een geldige cel heeft gekozen. Dat is nu zo, want er was nog niets ingevuld. result bevat dus de waarde true
5. De methode Board levert nu (bijvoorbeeld) de volgende representatie van het bord:

-------------

| O | 2 | 3 |

-------------

| 4 | 5 | 6 |

-------------

| 7 | 8 | 9 |

-------------

1. Status is nu op PlayerXPlays gezet, omdat het spel nog niet afgelopen is.
2. Kiest speler X nu bijvoorbeeld weer de cel 1,dan wordt de regel bool result = t.ChooseCell(1) nogmaals uitgevoerd en krijg result de waarde false. Status blijft op PlayerXPlays staan
3. Nu kiest speler X voor cel 2. Board levert nu de volgende representatie:

-------------

| O | X | 3 |

-------------

| 4 | 5 | 6 |

-------------

| 7 | 8 | 9 |

-------------

Status wordt nu weer op PlayerOPlays gezet.

1. Na een aantal beurten zal het spel gewonnen worden of gelijkspel gespeeld worden. In het eerste geval zal Status op PlayerOWins of PlayerXWins gezet worden. In het andere geval op Equal. De client kan nu bijvoorbeeld bepalen om een melding te tonen en daarna het spel automatisch te resetten, met de methode Reset.

**NB: bij de GUI-implementatie die gebruik maakt van TicTacToeEngine heb je in principe de methode Board niet nodig.**

1. Creeër de library met de class TicTacToeEngine.
2. Verander de GUI-implementatie uit opgave 1 zodanig dat deze gebruik maakt van TicTacToeEngine en geen spel-logica meer bevat.
3. Creeër een Console-applicatie waarin middels input van de gebruik het spel TicTacToe gespeeld kan worden. Maak weer gebruik van TicTacToeEngine. Hieronder zie je een voorbeeld van hoe de applicatie eruit zou kunnen komen te zien.

|  |  |
| --- | --- |
|  |  |

## Week 2 (Delegates, lambda expressies, events)

Deze opgave heeft als doel verder kennis te maken met delegates en lambda-expressies.

Lees hoofdstuk 5 van het boek nog eens rustig door en ga na of je de voorbeelden begrijpt.

Bekijk vervolgens de code in project practicum2 (Sharepoint) en vul het bestand LambdaRunner aan. Maak voor elke methode in het bestand Methods.cs een lambda-expressie in het bestand LambdaRunner en toon aan dat de output van de methoden dezelfde is als die van de lambda-expressies. Sla de lambda-expressies op in een lokale variabele van een type dat het meest logisch is voor het soort lambda: een Func, Action of Predicate.

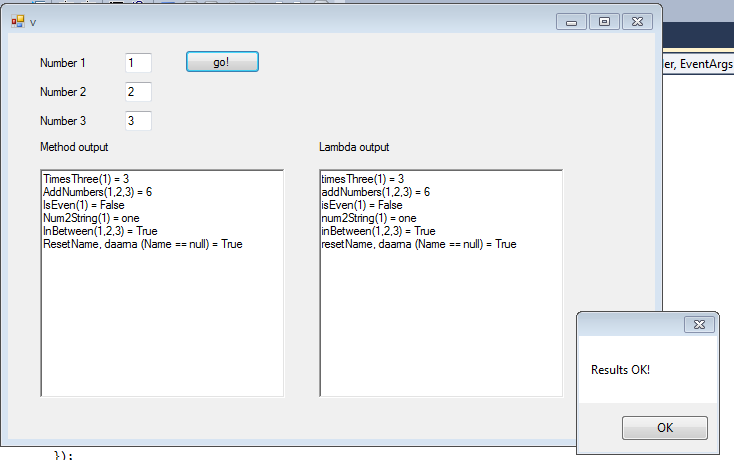
Voorbeeld. Voor de regels uit MethodRunner.cs:

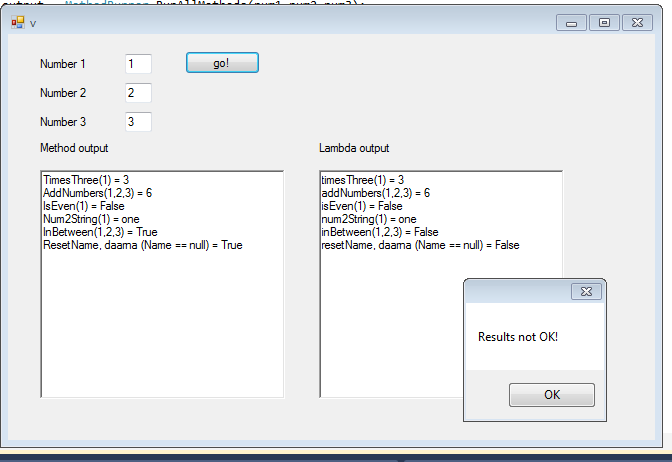
output.AppendFormat("TimesThree({0}) = {1}\n", num1, Methods.TimesThree(num1));

maak je in LambdaRunner.cs het volgende:

// methode TimesThree herschreven als lambda-expressie   
Func<int, int> timesThree = x => 3 \* x;   
output.AppendFormat("x => 3 \* x, met x={0} = {1}", num1, timesThree(num1));

1. Voeg functionaliteit toe aan het programma zodat er bij het opstarten drie random getallen worden ingevuld in de drie textboxex.
2. Voeg functionaliteit toe zodat automatisch gecontroleerd wordt of de waarden die na de =-tekens in de ene TextArea gelijk zijn aan de waarden na de =-tekens in de andere TextArea. In onderstaande afbeeldingen zie je daarvan twee voorbeelden: een wanneer de output gelijk is en de ander wanneer de output niet gelijk is.





e) Draai het programma met verschillende waarden in de drie textboxes en controleer of het programma naar behoren werkt.

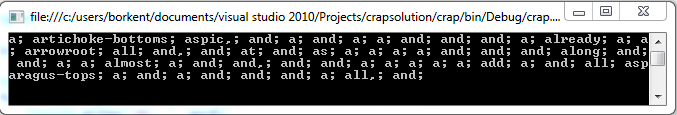
## Week 3

a) Schrijf een methode GetWords die woorden uit tekstbestand kan lezen die voldoen aan een op te geven conditie. In onderstaand voorbeeld worden alle woorden die beginnen met een a uitgelezen uit de textfile randomtext.txt (Sharepoint).

String path = @"c:\temp\randomtext.txt";

foreach (String word in GetWords(path, s => s.StartsWith("a")))

Console.Write("{0}; ", word);



De @ voor de string zorgt ervoor dat de \ niet als een escape-karakter wordt gezien. Zonder de @ zou je steeds tweemaal \ moeten gebruiken (\\).

De methode moet voldoen aan de volgende voorwaarden:

* Bij een ongeldig path moet er een zinvolle foutmelding op het scherm verschijnen
* Maak gebruik van yield return
* de methode heeft als return-type IEnumerable<String>

Raadpleeg hoofdstuk 11 voor het omgaan met files.

1. Haal met de methode GetWords alle woorden op die beginnen met de letter b en zet de IEnumerable om naar een Array van Strings. Sorteer de Array op lengte van laag naar hoog door middel van de methode Array.Sort en een lambda-expressie.

## Week 4

In dit practicum leer je werken met de XML-API zoals aangeboden in de namespace System.Xml.Linq (<http://msdn.microsoft.com/en-us/library/system.xml.linq.aspx>) in combinatie met de LINQ-syntax.

### Deel 1 XML creëren

1. Lees hoofdstuk 12, blz 449-452 (tot Creating XML Documents) en ga na of je begrijpt wat er met ‘element’ en ‘hierarchy’ wordt bedoeld.
2. Ga verder met Creating XML Documents blz 452-556 en bestudeer de bijbehorende code. De code staat ook op SharePoint en kan je overnemen in Visual Studio om er mee te spelen.
3. Maak nu een nieuw project met daarin de class Track en CD. Een CD kan bestaan uit meerdere Tracks en heeft een titel en een artiest. Een Track heeft een titel, een artiest en een lengte. Maak voor de lengte gebruik van de class TimeSpan uit de namespace System.
4. Creeër vervolgens een object van class CD. Geef de CD een naam en voeg drie tracks toe. De artiest-, titel- en trackinformatie moet van een echt album afkomstig zijn. In deel 2 van het practicum gaan we namelijk de overige tracks automatisch aanvullen via een webservice van Last.FM.
5. Transformeer nu het CD-object uit vraag 4 naar XML en toon de XML op het scherm. Maak gebruik van de XML-API uit System.Linq.Xml zoals je in de code die je bij vraag 2 ook gezien hebt. Denk eraan om de lengte naar een String te converteren. Gebruik een foreach-loop om de Track-elementen aan te maken en toe te voegen aan zijn parent-element. De XML moet er dan als volgt uit komen te zien (met de artiest, naam, titels, etc. van het door jouw gekozen album):

<cd>

<artist>Dream Theater</artist>

<name>Awake</name>

<tracks>

<track>

<artist>Dream Theater</artist>

<title>6:00</title>

<length>00:05:31</length>

</track>

<track>

<artist>Dream Theater</artist>

<title>Caught in a Web</title>

<length>00:05:28</length>

</track>

<track>

<artist>Dream Theater</artist>

<title>Innocence Faded</title>

<length>00:05:34</length>

</track>

</tracks>

</cd>

1. Lees pagina 456 t/m 459 (tot Putting the LINQ in LINQ to XML).

* Verander de naam en artiest van de CD in een attribuut
* Herschrijf het programma zodat je in de foreach-loop, waarin je de track-elementen creëert, de kortere manier van XML construeren gebruikt uit Example 12\_4 (Constructing an XElement all at once op blz 459).

De XML ziet er dan weer hetzelfde uit als hierboven, met uitzondering van artist en name, welke nu attributen zijn geworden van het cd-element:

<cd artist="Dream Theater" name="Awake">  
 <tracks>...

1. Lees nu verder op blz 459 t/m 461 (tot Searching in XML with LINQ). In Example 12\_5 wordt nu met één enkel statement de gehele XML-structuur aangemaakt. Pas dit ook toe op de XML die bij de CD hoort.

### Deel 2 XML parsen/doorzoeken

In dit gedeelte maken we gebruik van de API van Last.FM. Last.FM is een service waarbij je kunt bijhouden welke muziek jij afspeelt. Op basis van jouw afgespeelde muziek doet Last.FM suggesties voor muziek die jij eventueel ook leuk zou vinden. Last.FM ondersteunt een REST API waarbij je gegevens van Last.FM kunt lezen en schrijven.

Normaal gesproken moet je als developer zelf een API-key aanmaken, maar speciaal voor dit practicum is er al eentje aangemaakt. Deze key moet je meegeven op de URL . De API-key voor dit practicum is: b5cbf8dcef4c6acfc5698f8709841949.  
  
We maken gebruik van de REST-service van Last.FM. Alle benodigde info daarover kun je vinden op: <http://www.last.fm/api/rest>.

1. Haal via een browser de informatie van het album op waarvan je bij deel 1 in het practicum al enige tracks hebt ingevuld. Maak gebruik van de REST API van Last.FM en gebruik de methode album.getinfo. Je krijgt in de browser het resultaat in de vorm van XML te zien. Een tip voor Chrome-gebruikers: installeer de XML Tree extensie, deze laat XML op een leesbare manier zien. De URL waarmee je informatie op kunt halen voor het album Awake van de artiest Dream Theater:   
   <http://ws.audioscrobbler.com/2.0/?method=album.getInfo&album=awake&artist=Dream%20Theater&api_key=b5cbf8dcef4c6acfc5698f8709841949>
2. Nu we weten via welke URL je de XML op kunt halen, gaan we dit in C# doen. Neem daarvoor het volgende code-fragment over in Visual Studio en vul op de puntjes in de URL de juiste gegevens in. Test de werking van de code, door bijvoorbeeld een breakpoint te zetten na de regel met het XDocument. Je kunt dan, nadat je het programma hebt opgestart, met de muis boven myXMLDoc gaan staan en inspecteren of de variabele de juist inhoud heeft gekregen.

String xmlString;

using (WebClient wc = new WebClient())

{

xmlString = wc.DownloadString(@"http://ws.audioscrobbler.com/2.0/?method=album.getinfo&...&api\_key=b5cbf8dcef4c6acfc5698f8709841949");

}

XDocument myXMLDoc = XDocument.Parse(xmlString);

}

Toelichting.

Voor gebruik van het using statement op bovenstaande manier zie: <http://msdn.microsoft.com/en-us/library/yh598w02(v=vs.80).aspx>. In het kort komt het erop neer dat resources, zoals file handles, netwerkconnecties, etc. meteen opgeruimd worden bij het verlaten van het using-block. Dit is beter dan wanneer je het de garbage-collector laat doen, omdat je dan niet weet wanneer de gebruikte resource wordt vrijgegegeven.

Een WebClient is een facade-class voor de HTTPWebRequest class. Deze laatste geeft je meer controle over bijvoorbeeld het zetten van cookies, maar voor ons doel is de WebClient hier voldoende. Zoals je ziet kan je hiermee eenvoudig wat data ophalen van het web.

1. Lees blz 461-466 door om vertrouwd te raken met het zoeken in XML met behulp van LINQ.
2. Print alle tracks op het scherm die **niet** in het CD-object van vraag 4, deel 1 staan. Gebruik hiervoor een geneste query. In de buitenste query doorzoek je de tracks en in de binnenste query controleer je of de track wel of niet in het CD-object zit. Vergelijk alleen op artiest en naam van de track. De buitenste query levert een verzameling van Track-objecten op die je vervolgens in een for-each loop kan afdrukken.
3. Pas de vorige opgave aan zodat je het CD-object aanvult met de tracks uit de vorige vraag.
4. Gebruik weer hetzelfde statement om XML te genereren uit het CD-object als bij vraag 7 uit deel 1.
5. Toon de XML op het scherm. De XML ziet er dan bijvoorbeeld zo uit:

<cd artist="Dream Theater" name="Awake">

<tracks>

<track>

<artist>Dream Theater</artist>

<title>6:00</title>

<length>00:05:31</length>

</track>

<track>

<artist>Dream Theater</artist>

<title>Caught in a Web</title>

<length>00:05:28</length>

</track>

<track>

<artist>Dream Theater</artist>

<title>Innocence Faded</title>

<length>00:05:34</length>

</track>

<track>

<artist>Dream Theater</artist>

<title>A Mind Beside Itself: Erotomania</title>

<length>00:06:44</length>

</track>

<track>

<artist>Dream Theater</artist>

<title>A Mind Beside Itself: Voices</title>

<length>00:09:53</length>

</track>

<track>

<artist>Dream Theater</artist>

<title>A Mind Beside Itself: The Silent Man</title>

<length>00:03:47</length>

</track>

<track>

<artist>Dream Theater</artist>

<title>The Mirror</title>

<length>00:08:15</length>

</track>

<track>

<artist>Dream Theater</artist>

<title>Lie</title>

<length>00:06:33</length>

</track>

<track>

<artist>Dream Theater</artist>

<title>Lifting Shadows Off a Dream</title>

<length>00:06:05</length>

</track>

<track>

<artist>Dream Theater</artist>

<title>Scarred</title>

<length>00:10:59</length>

</track>

<track>

<artist>Dream Theater</artist>

<title>Space-Dye Vest</title>

<length>00:07:29</length>

</track>

</tracks>

</cd>

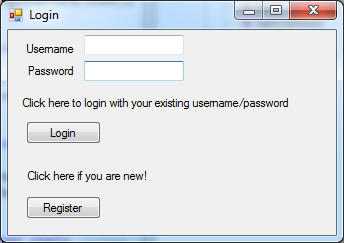
## Week 5 en 6

Maak een webservice en client-applicatie(s) voor een winkel met producten. Een product heeft een naam en een prijs. De winkel heeft van elk product een beperkt aantal exemplaren in huis. Een klant moet met een client-applicatie de volgende zaken kunnen doen en het systeem moet voldoen aan de terloops genoemde eisen:

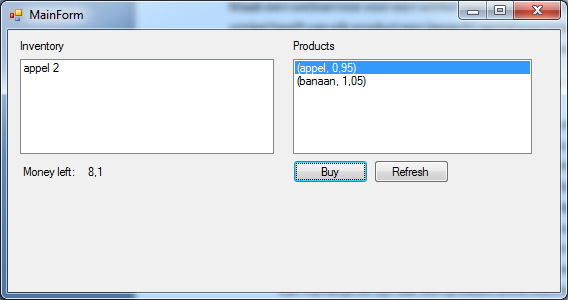
* Registreren. Bij het opgeven van een username krijgt de klant een gegenereerd paswoord terug (bijvoorbeeld zijn naam omgekeerd), mits de username nog niet in gebruik is
* Inloggen.
* Een lijst van productgegevens ophalen (in elk geval hun namen en prijzen, mogelijk ook hun aantallen). Deze lijst moet automatisch worden getoond nadat de klant is ingelogd. Ook moet er de mogelijkheid zijn om deze lijst met een druk op de knop te verversen. Het kan namelijk zo zijn dat een product uitverkocht is geraakt, doordat een andere klant een product heeft gekocht. Dan moet het product niet meer getoond worden. Andersom kan het zo zijn dat de winkel de voorraad heeft aangevuld. Dan moet het product juist wel weer in het lijstje verschijnen.
* Een product kopen. Dit kan alleen als het product nog voldoende op voorraad is en de klant nog voldoende saldo heeft.
* Een lijst van door de klant gekochte producten ophalen en hun hoeveelheden.
* Saldo-informatie ophalen. Bij het registreren krijgt de klant automatisch een tegoed waarmee hij een paar producten kan kopen.

Om het simpel te houden maken we gebruik van BasicHTTPBinding zodat je met wsimport een Java-proxy kan genereren.   
Je mag aan elke webservice-methode een username/password-combinatie meegeven die op de server worden gecontroleerd. Je zou daarnaast aan de webservice een methode kunnen toevoegen die test of een username/password-combinatie geldig is. Met behulp van deze methode kun je de klant laten inloggen en in de rest van het proces de ingevoerde gegevens telkens hergebruiken.   
Uiteraard moet de state van de webservice gedeeld zijn voor alle clients. Dat wil zeggen dat als een product voor de ene client is uitverkocht, dit uiteraard ook voor een andere client geldt.

Ter illustratie de volgende schermafbeeldingen van een mogelijke client-applicatie:







De practicumopdracht maakt deel uit van het eindcijfer.

Er wordt op de volgende punten beoordeeld:

* De webservice voldoet aan alle genoemde eisen. Maximaal 40 punten.
* Er is een .NET-client met goed ogende user interface gebouwd welke naar behoren werkt vanuit het oogpunt van de gebruiker, inclusief foutafhandeling. Maximaal 10 punten.
* De .NET-client is gebouwd met WPF-technologie (niet behandeld in de les). Maximaal 10 punten.
* Er is een Java-client gebouwd. Een volledig werkende console-versie levert maximaal 10 punten op, een GUI-client (bijvoorbeeld met Swing) maximaal 15 punten.
* De programmeurs hebben in hun C#-code de concepten uit het vak .NET Programmeren juist toegepast. Maximaal 15 punten.
* Er is juist persistentie met SQL Server toegepast op de server. Maximaal 10 punten.

Laat bij de demonstratie van je eindproduct ook de volgende zaken zien:

* Een klant heeft na het kopen van een aantal producten te weinig saldo en krijgt hiervan een nette melding.
* Een product raakt uitverkocht waardoor het verdwijnt uit de lijst van beschikbare producten op verschillende clients (eventueel na het drukken op een refresh-knop).
* Er kunnen dus tussen de 0 en 100 punten gehaald worden.