

2011

Opdrachten Object georiënteerd programmeren met C#



Stichting Praktijkleren
ROC van Twente
22-9-2011

Inhoud

Opdracht 1 - De *Track* Class..... 4

 Stap 1: Nieuw project aanmaken 4

 Stap 2: Private fields toevoegen..... 4

 Stap 3: Constructors toevoegen 6

 Stap 4: Methods toevoegen 6

 Stap 5: Properties toevoegen..... 6

 Stap 6: Track Class testen 7

Opdracht 2- De *TrackList* Class..... 12

 Stap 1: Nieuwe Class aanmaken..... 12

 Stap 2: Private fields toevoegen..... 12

 Stap 3: Constructors toevoegen..... 12

 Stap 4: Methods toevoegen 12

 Stap 5: Properties toevoegen..... 13

 Stap 6: TrackList Class testen 14

Opdracht 3 - De *AudioDevice* Class 17

 Stap 1: Nieuwe Class aanmaken..... 17

 Stap 2: Private fields toevoegen..... 18

 Stap 3: Methods toevoegen 18

 Stap 4: Properties toevoegen..... 19

Opdracht 4, De *MemoRecorder* Class..... 20

 Stap 1: Nieuwe Class aanmaken..... 20

 Stap 2: Private field toevoegen 20

 Stap 3: Constructor toevoegen 20

 Stap 4: Method toevoegen..... 21

 Stap 5: Property toevoegen 21

 Stap 6: MemoRecorder Class testen 21

Opdracht 5, De *CdDiscMan* Class 24

 Stap 1: Nieuwe Class aanmaken..... 24

 Stap 2: Interface *IDisplay* toevoegen..... 24

 Stap 3: Private fields toevoegen..... 25

 Stap 4: Constructor toevoegen 25

 Stap 5: Methods toevoegen 25



Stap 6: Properties toevoegen	26
Stap 7: CdDiscMan Class testen	26
Opdracht 6, De <i>Mp3Player</i> Class	28
Stap 1: Nieuwe Class aanmaken.....	28
Stap 2: Interface <i>ITrackList</i> toevoegen	28
Stap 3: Private fields toevoegen.....	28
Stap 4: Constructor toevoegen	29
Stap 5: Methods toevoegen	29
Stap 6: Properties toevoegen.....	30
Stap 7: <i>Mp3Player</i> Class testen	30

Opdracht 1 - De Track Class

SoundSharp wil de software die ze ontwikkelt voor intern gebruik verder professionaliseren. Besloten is om een centrale Class Library te maken met veel gebruikte Classes. Deze Class Library kan vervolgens bij elke nieuwe applicatie gebruikt worden.

De eerste Class die ontwikkeld wordt is de Track Class. Deze Class stelt een losse MP3 track voor.

Stap 1: Nieuw project aanmaken

Maak in Visual Studio een nieuw project aan met de naam AudioDevices, kies hiervoor de Class Library template. Rename in de solution explorer Class1.cs naar Track.cs in het geval je voor C# hebt gekozen of Class1.vb naar Track.vb als je voor VB.Net hebt gekozen.

Verander de namespace definitie in:

```
namespace AudioDevices.Tracks
```

Verander de naam van de Class in: Track.

Stap 2: Private fields toevoegen

Voeg in de Track Class de volgende private fields toe:

Naam	Type	Omschrijving
id	integer	Unieke numerieke id voor de track.
name	string	Naam van de track.
artist	string	Naam van de uitvoerende artiest.
albumSource	string	Naam van het album waar de track van afkomstig is.
style	category*	Muziek style categorie.
length	time**	Lengte van de track.

Category type*

Definieer een enumerator met de naam Category om een lijst van muziekstijlen op te slaan. De enumerator moet de volgende waarden bevatten:

```
Ambient
Blues
Country
Disco
Electro
Hardcore
HardRock
HeavyMetal
Hiphop
Jazz
Jumpstyle
Klassiek
Latin
Other
Pop
Punk
Reggae
Rock
Soul
Trance
Techno
```

Time type**

Om de lengte van een track op te slaan maak je gebruik van een structure Time. Maak deze structure met de volgende specificaties:

- De structure moet drie integer variabelen bevatten: hours, minutes en seconds.
- De structure moet de volgende drie constructors bevatten:

```
public Time(int seconds)
public Time(int minutes, int seconds)
public Time(int hours, int minutes, int seconds)
```

Zorg ervoor dat de constructor de juiste waarden toekent aan de interne variabelen.

Voorbeelden:

```
Time t = new Time(100)
```

Nu dient de structure t de volgende waarden te bevatten:

```
hours:      0
minutes:    1
seconds:    40
```

```
Time t = new Time(150,45)
```

Nu dient de structure t de volgende waarden te bevatten:

hours: 2
minutes: 30
seconds: 45

minutes en seconds mogen dus nooit een waarde te bevatten hoger dan 59.

Override de ToString() method van de Time structure zodat deze een string terug geeft met de volgende format: <hours> : <minutes> : <seconds>.

Stap 3: Constructors toevoegen

Voeg aan de Track Class de volgende constructors toe:

```
public Track()  
public Track(int id)  
public Track(int id, string name)  
public Track(int id, string artist, string name)
```

Zorg ervoor dat binnen de constructors de opgegeven waarden in de constructor parameters gebruikt worden om de private fields te initialiseren.

Stap 4: Methods toevoegen

De Track Class dient de volgende twee public methods te bevatten:

```
GetLength()
```

De method GetLength() geeft de lengte van de Track terug in de vorm van een Time structure.

```
GetLengthInSeconds()
```

De method GetLengthInSeconds() geeft de lengte van de Track in secondes terug in de vorm van een integer variabele. In deze method vindt dus een conversie plaats van de tijd, geregistreerd in de private field length van het type structure Time, naar een integer.

Stap 5: Properties toevoegen

Voeg in de Track Class de volgende properties toe. De kolommen Get en Set geven aan of de property respectievelijk een getter en/of een setter dient te bevatten.

Naam	Type	Get	Set	Omschrijving
Id	integer	✓	✓	Unieke numerieke id voor de track.
Name	string	✓	✓	Naam van de track.

Artist	string	✓	✓	Naam van de uitvoerende artiest.
DisplayName*	string	✓	✗	Naam van de uitvoerende artiest en naam van de track.
Length	time	✗	✓	Lengte van de track.
DisplayLength	string	✓	✗	Lengte van de track als een string.
Style	category	✓	✓	Muziek style categorie.
AlbumSource	string	✓	✓	Naam van het album waar de track van afkomstig is.

DisplayName*

DisplayName toont alleen de naam van de uitvoerende artiest en track als deze beiden een waarde hebben. Als dat niet het geval is dient de tekst: "unknown" getoond te worden.

Stap 6: Track Class testen

De Track Class is nu compleet. Om deze te testen maken we een nieuwe Windows applicatie aan in dezelfde Visual Studio solution.

Kies File | New Project... en maak een nieuwe Windows Forms Application aan met de naam Tester. Zorg ervoor dat dit project bij de bestaande solution wordt toegevoegd (Solution: Add to Solution). Maak nu binnen het Tester project als volgt een referentie naar de AudioDevices Class Library:

- Selecteer References en vervolgens uit het context menu (rechtermuis klik) Add Reference.
- Ga naar de Projects tab.
- Selecteer project AudioDevices en vervolgens [ok].

Verander de namespace in Tester en voeg de AudioDevices.Tracks namespace toe.

```
using AudioDevices.Tracks;
```

De verschillende opdrachten gaan we maken in afzonderlijke formulieren. Om er één geheel van te maken beginnen we met het maken van een hoofdmenu, van waaruit we de later aan te maken formulieren kunnen openen.

Voeg nu eerst een nieuwe Form aan het project toe en wijzig de naam in **Main**. Stel de volgende eigenschappen van het formulier in:

Size	Hoogte 600 pixels en breedte 800 pixels.
FormBorderStyle	FixedDialog
StartPosition	CenterScreen
Text	Hoofdmenu Soundsharp

Voeg aan het formulier een *menuStrip* toe met de volgende menustructuur:

- Bestand
 - Afsluiten
- Track
 - Nieuw
 - Wijzig

Voeg ook het logo toe aan het menu en dock dit aan de onderkant van het formulier. Plaats tenslotte een label op het formulier met een tekst die het doel van de applicatie aangeeft.

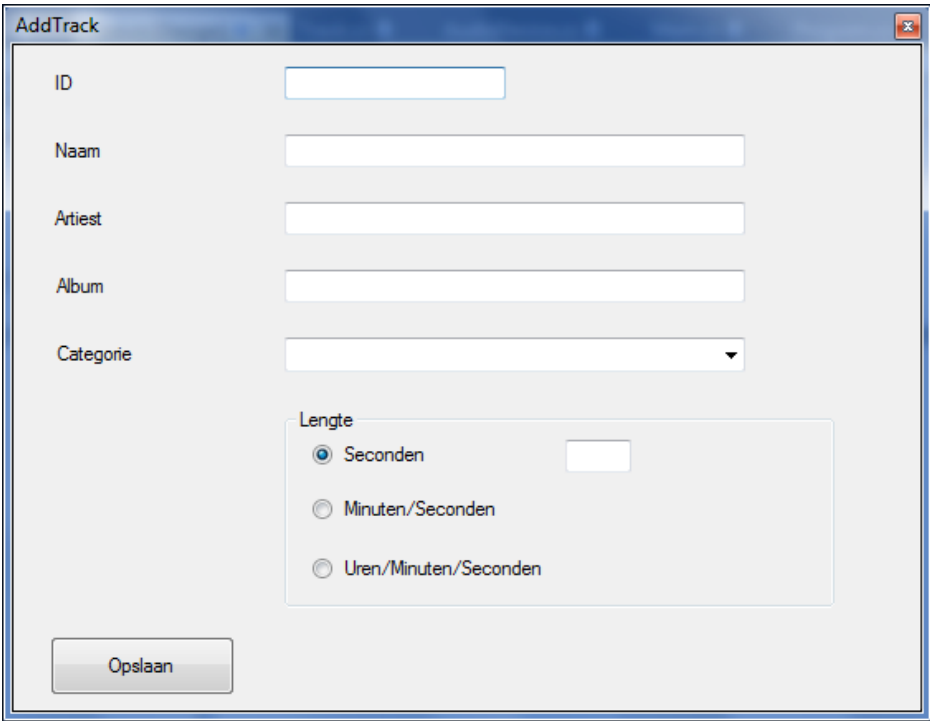
De uiteindelijke layout zou er dan ongeveer zo uit kunnen zien (zie Figuur 1):



Figuur 1

Vervolgens gaan we een formulier maken waarmee de gebruiker een nieuwe track kan invoeren. Dit formulier wordt in het hoofdmenu aangeroepen vanuit Track->Nieuw.

Een mogelijke layout van het formulier kan er als volgt uit komen te zien (zie **Fout! Verwijzingsbron niet gevonden.**):



Figuur 2

Het formulier kent o.a. de volgende controls:

txtID	Voor het invoeren van de ID
txtName	Voor het invoeren van de titel van de track
txtArtist	Voor het invoeren van de uitvoerende artiest
txtAlbumSource	Voor het invoeren van de titel van het album waar track vanaf komt
cbCategory	Voor het selecteren van één van de beschikbare categorieën. Deze combobox moet tijdens het laden van het formulier gevuld worden met alle items uit de enum category.
gbLengte	Een groepbox met daarop 3 radiobuttons, zodat de gebruiker op 3 verschillende manieren de lengte van de track kan invoeren. De radiobuttons corresponderen met de 3 constructors van de struct Time. De eerste radiobutton toont de gebruiker 1 invoervak, de tweede constructor toont 2 vakken en de derde constructor toont er 3.
btnAddTrack	Maakt een nieuwe instantie aan van de track op basis van de door de gebruiker ingevoerde gegevens.

Voeg nu code toe aan het click-event van de button btnAddTrack en controleer of de Track Class goed functioneert door aan het eind van de code een messagebox te tonen die de volgende gegevens laat zien van de track:

- DisplayName
- DisplayLength
- Tijd in seconden (m.b.v. de method GetLengthInSeconds)
- Categorie

Nadat de messagebox is getoond moeten alle tekstvakken leeggemaakt te worden, de eerste radiobutton moet weer geselecteerd worden. De geselecteerde waarde in de combobox mag blijven staan.

Opdracht 2- De *TrackList* Class

De volgende Class die ontwikkeld wordt is de *TrackList* Class. Deze Class stelt een lijst met tracks voor.

Stap 1: Nieuwe Class aanmaken

Open in Visual Studio de *AudioDevices* solution. Selecteer in de solution explorer het *AudioDevices* project en vervolgens uit het context menu (rechtermuis klik) *Add | New item...* Kies in de template dialog de template *Class* en geef deze de naam *TrackList.cs* in het geval je voor C# hebt gekozen of *TrackList.vb* als je voor VB.NET hebt gekozen.

Verander de namespace definitie in:

```
namespace AudioDevices.Tracks
```

Verander de naam van de Class in: *TrackList*.

Stap 2: Private fields toevoegen

De *TrackList* Class bevat een lijst met *Track* objecten. Deze lijst is het enige private field van de *TrackList* Class. Voor de implementatie maken we gebruik van de generic .NET Class *List<>*. Deze Class bevindt zich in de namespace *System.Collections.Generic*. Voeg deze namespace toe aan de *TrackList* Class

```
using System.Collections.Generic;
```

Voeg nu het private field *tracks* toe van het type *List<Track>*.

```
Private List<Track> tracks;
```

Stap 3: Constructors toevoegen

Voeg aan de *Track* Class de volgende constructors toe:

```
public TrackList()  
public TrackList(List<Track> tracks)
```

Binnen de constructor zonder parameters dient het private field *tracks* met een lege instantie geïnitieerd te worden.

In de tweede constructor geeft men een gevulde lijst met *Track* objecten als parameter mee. Deze lijst dient dan aan het private field *tracks* toegekend te worden.

Stap 4: Methods toevoegen

De *TrackList* Class dient de volgende public methods te bevatten:

Add (...)

```
public void Add(Track t)
```

De Method *Add(...)* accepteert een *Track* object als parameter en voegt deze *Track* vervolgens toe aan de interne lijst met *tracks*.

Tip: De generic Class List<> bevat zelf al een Add method waar je gebruik van kan maken.

Remove (...)

```
public void Remove (Track t)
```

De Method Remove(...) accepteert een Track object als parameter en verwijdert deze Track vervolgens van de interne lijst met tracks.

Tip: De generic Class List<> bevat zelf al een Remove method waar je gebruik van kan maken.

Clear ()

De method Clear() verwijdert alle toegekende tracks van de interne lijst met tracks.

Tip: De generic Class List<> bevat zelf al een Clear method waar je gebruik van kan maken.

GetAllTracks ()

De method GetAllTracks() levert de lijst op met tracks van het tracklist object van het type:

```
List<Track>
```

Shuffle ()

De method Shuffle() levert een lijst op met tracks van het tracklist object van het type:

```
List<Track>
```

Deze lijst dient een random (willekeurig verdeelde) versie te zijn van het private field tracks. Het is echter de bedoeling dat de volgorde van de tracks in het private field tracks niet veranderd wordt als men Shuffle() aanroept.

Tips:

- Maak gebruik van de .NET Class Random.
- Maak gebruik van een kopie van het private fields tracks om ervoor te zorgen dat de interne volgorde van tracks niet veranderd.

Stap 5: Properties toevoegen

Voeg in de TrackList Class de volgende properties toe. De kolommen Get en Set geven aan of de property respectievelijk een getter en/of een setter dient te bevatten.

Naam	Type	Get	Set	Omschrijving
Count	integer	✓	✗	Aantal tracks in de tracklist
TotalTime*	time	✓	✗	Totale tijd van de tracklist

TotalTime*

Om de totale tijd te berekenen van alle tracks in de tracklist, kan je het beste met een loop de track objecten die de tracks variabele bevat, doorlopen:

```
foreach (Track track in tracks)
```

M.b.v. de Track method GetLength() kun je van elke track de duur opvragen en met behulp van interne variabelen de totale tijd bijwerken.

Stap 6: TrackList Class testen

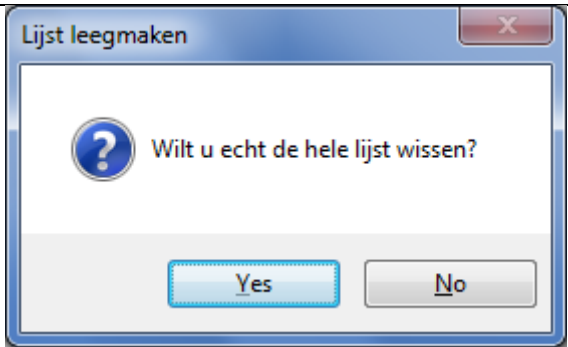
De TrackList Class is nu compleet.

Breid nu het project **Tester** zodanig aan dat de door de gebruiker toegevoegde tracks opgeslagen worden in een tracklist.

Voeg aan static TrackList toe aan het formulier Main. Zorg dat bij het starten van de applicatie automatisch de volgende tracks worden toegevoegd:

```
Track t1 = new Track(1, "Prince", "Guitar");  
t1.Length = new Time(4, 12);  
  
Track t2 = new Track(2, "Nelly Furtado", "Say it Right");  
t2.Length = new Time(4, 41);  
  
Track t3 = new Track(3, "David Guetta & Chris Willis", "Love is gone");  
t3.Length = new Time(3, 50);
```

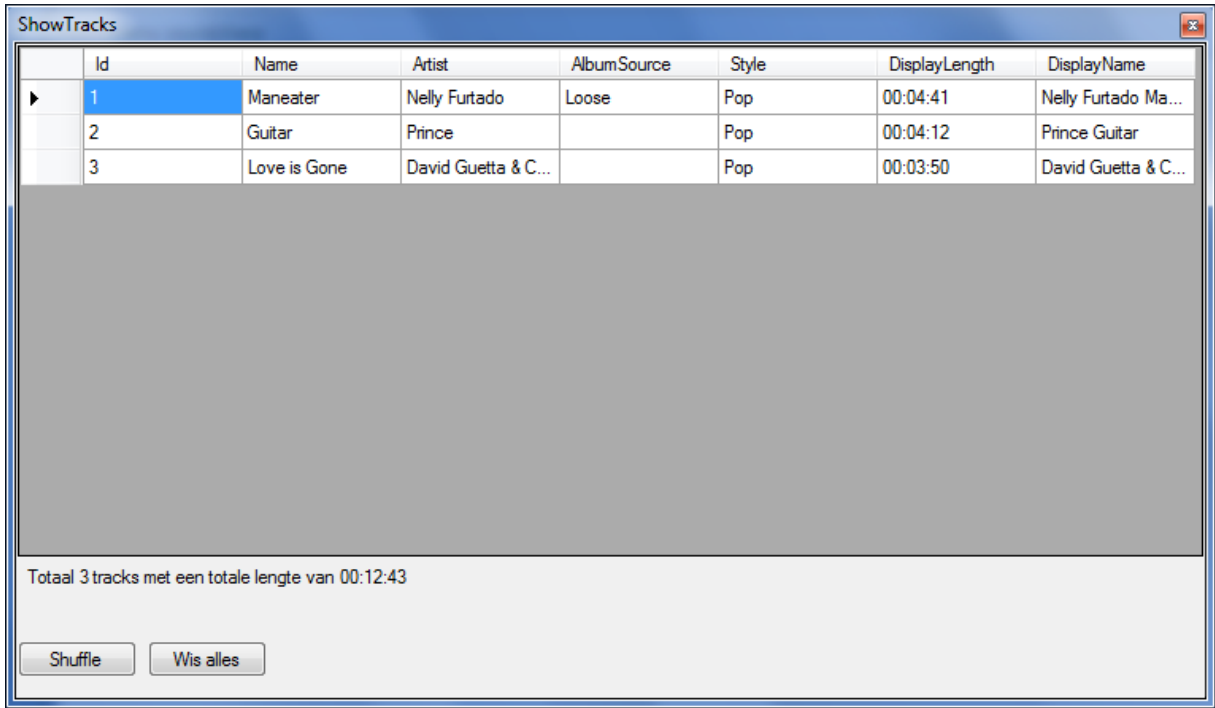
Voeg aan de menuoptie Tracks van het formulier Main een optie Toon toe. Als deze optie wordt geselecteerd dient een nieuw formulier getoond te worden met een DataGridView, waarin alle tracks van de tracklist worden getoond. Ook dient een knop toegevoegd te worden die ervoor zorgt dat de lijst geshuffled kan worden. Voeg tevens een knop toe die de gebruiker in staat stelt de hele lijst te wissen, echter niet nadat middels een messagebox om een bevestiging is gevraagd (zie voor een voorbeeld **Figuur 3**).



Figuur 3

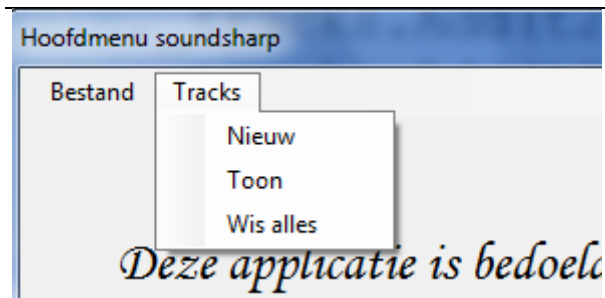
Voeg aan de menuoptie Tracks van het formulier Main eenzelfde optie toe om de hele lijst met tracks te wissen. Ook nu moet om een bevestiging worden gevraagd.

De layout van dat formulier kan er ongeveer uit gaan zien als in **Figuur 4**.



Figuur 4

Het aangepaste menu van het formulier Main ziet er ongeveer uit als in **Figuur 5**.

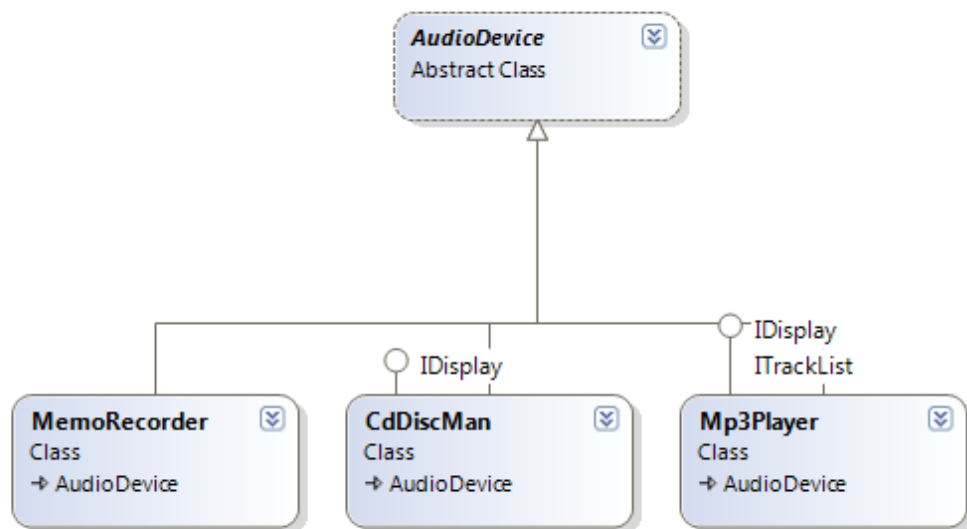


Figuur 5

Opdracht 3 - De *AudioDevice* Class

De volgende Class die ontwikkeld wordt is de *AudioDevice* Class. Deze Class stelt een apparaat voor dat muziek af kan spelen. Dat kan een mp3 speler zijn maar ook een radio of een walkman bijv. In latere opdrachten gaan we classes aanmaken voor de specifieke audiodevices die SoundSharp in het assortiment heeft.

Om alvast een idee te geven van wat we uiteindelijk gaan maken is in Figuur 6 een classdiagram weergegeven van de diverse classes die gemaakt gaan worden.



Figuur 6

De *AudioDevice* Class definiëren we als een abstracte Class dat wil zeggen dat we van deze Class geen objecten kunnen aanmaken. We gebruiken deze abstract Class *AudioDevice* later om er concrete classes van af te leiden.

Stap 1: Nieuwe Class aanmaken

Open in Visual Studio de *AudioDevices* solution. Selecteer in de solution explorer het *AudioDevices* project en vervolgens uit het context menu (rechtermuis klik) Add | New item... Kies in de template dialog de template Class en geef deze de naam *AudioDevice.cs* in het geval je voor C# hebt gekozen of *AudioDevice.vb* als je voor VB.NET hebt gekozen.

Verander de namespace definitie in:

```
namespace AudioDevices.Devices
```

Verander de naam van de Class in: *AudioDevice* en zorg ervoor dat je de Class abstract maakt. Gebruik hiervoor het keyword **abstract**.

Stap 2: Private fields toevoegen

Voeg in de AudioDevice Class de volgende private fields toe:

Naam	Type	Omschrijving
serialId	integer	Unieke numerieke serialId van het device. Wordt automatisch bepaald.
model	string	Modelnaam van het device.
make	string	Merk van het device.
priceExBtw	decimal	Prijs van het device exclusief BTW.
creationDate	datetime	Datum/Tijd waarop het device gemaakt is.
btwPercentage*	double	BTW Percentage.

btwPercentage*

BtwPercentage is een static readonly field en wordt als volgt gedefinieerd:

```
protected readonly static double btwPercentage = 19.0;
```

NB. Omdat AudioDevice een abstract Class is kunnen er geen objecten van afgeleid worden er hoeven dan ook geen constructors gedefinieerd te worden.

Stap 3: Methods toevoegen

De AudioDevice Class dient de volgende public methods te bevatten:

DisplayIdentity()

De DisplayIdentity() method levert identity informatie in de vorm van een string op. Als de method aangeroepen wordt zonder parameters wordt de tekst "Serial: [serialId]" terug gegeven.

Maak ook een overload van DisplayIdentity met twee boolean parameters: makeInfo en modelInfo. Als modelInfo **true** is wordt aan de string de tekst " Make: [make]" toegevoegd. Als makeInfo **true** is wordt aan de string de tekst " Model: [model]" toegevoegd.

GetDeviceLifeTime()

De GetDeviceLifeTime() method levert informatie op over het aantal dagen dat het device oud is. Deze informatie wordt teruggeven als een string.

Als het veld creationDate gevuld is wordt het verschil in dagen tussen de huidige datum en de creationDate berekend. Dit verschil wordt als de volgende string terug gegeven:

```
"Lifetime [verschil] days"
```

Als het veld creationDate niet gevuld is wordt de string "Lifetime unknown" terug gegeven.

Tip: Je kunt de .NET Class TimeSpan gebruiken om een verschil tussen twee data op te slaan.

Voorbeeld:

```
TimeSpan diff = DateTime.Now.Subtract(this.creationDate);
```

DisplayStorageCapacity()

De method DisplayStorageCapacity() is een abstract method en heeft dus binnen de AudioDevice Class geen implementatie. Deze method levert een string op met informatie over de opslagcapaciteit van het device. Definieer DisplayStorageCapacity() als volgt:

```
public abstract string DisplayStorageCapacity();
```

Stap 4: Properties toevoegen

Voeg in de AudioDevice Class de volgende public properties toe. De kolommen Get en Set geven aan of de property respectievelijk een getter en/of een setter dient te bevatten.

Naam	Type	Get	Set	Omschrijving
SerialId	integer	✓	✗	Unieke numerieke serialId van het device. Wordt automatisch bepaald.
Model	string	✓	✓	Modelnaam van het device.
Make	string	✓	✓	Merk van het device.
PriceExBtw	decimal	✓	✓	Prijs van het device exclusief BTW.
ConsumerPrice	decimal	✓	✗	Prijs van het device inclusief BTW.
CreationDate	date/time	✓	✓	Datum/Tijd waarop het device gemaakt is.

NB. Omdat AudioDevice een abstract Class is en er geen objecten van afgeleid kunnen worden test je de Class nu niet. Dit gebeurt in de Classes die je van AudioDevice afleidt.

Opdracht 4, De *MemoRecorder* Class

De volgende Class die ontwikkeld wordt is de MemoRecorder Class. Deze Class stelt een memocorder voor waarmee met behulp van cartridges gesprekken opgenomen kunnen worden.

De MemoRecorder Class definiëren we als een concrete Class, d.w.z. dat we van deze Class objecten kunnen aanmaken. De MemoRecorder Class is een derived Class van base Class AudioDevice.

Stap 1: Nieuwe Class aanmaken

Open in Visual Studio de AudioDevices solution. Selecteer in de solution explorer het AudioDevices project en vervolgens uit het context menu (rechtermuis klik) Add | New item... Kies in de template dialog de template Class en geef deze de naam MemoRecorder.cs in het geval je voor C# hebt gekozen of MemoRecorder.vb als je voor VB.NET hebt gekozen.

Verander de namespace definitie in:

```
namespace AudioDevices.Devices
```

Verander de naam van de Class in MemoRecorder en zorg ervoor dat deze Class van AudioDevice is afgeleid.

Stap 2: Private field toevoegen

Voeg in de MemoRecorder Class de volgende private field toe:

Naam	Type	Omschrijving
maxCartridgeType	MemoCartridgeType*	Cartridge met de grootste opslagcapaciteit die gebruikt kan worden bij deze memorecorder.

MemoCartridgeType*

Definieer een enumerator met de naam MemoCartridgeType om een lijst van beschikbare Memo cartridge types op te slaan. De enumerator moet de volgende waarden bevatten:

```
C60  
C90  
C120  
Anders
```

Stap 3: Constructor toevoegen

Voeg aan de Class de volgende constructor toe:

```
public MemoRecorder()
```

Zorg ervoor dat het protected base field serialId geïnitieerd wordt met de waarde van de laatst gebruikte serialId verhoogd met 1. Daarvoor voeg je aan de class AudioDevice een static variabele toe waarin de laatst gebruikte ID wordt opgeslagen.

```
//Varibale waarin de laatst gebruikte serialID wordt bewaard.  
protected static int lastID = 1;
```

Stap 4: Method toevoegen

De MemoRecorder Class moet een implementatie bevatten van de abstract method

DisplayStorageCapacity()

DisplayStorageCapacity() levert informatie over de opslagcapaciteit van de memorecorder op in een string.

De method maakt hiervoor gebruik van de waarde van het private field: maxCartridgeType.

Waarde maxCartridgeType	Return value
C60	"Max capacity 60 min."
C90	"Max capacity 90 min."
C120	"Max capacity 120 min."
Anders	"Max capacity unknown"

Stap 5: Property toevoegen

Voeg in de MemoRecorder Class de volgende property toe. De kolommen Get en Set geven aan of de property respectievelijk een getter en/of een setter dient te bevatten.

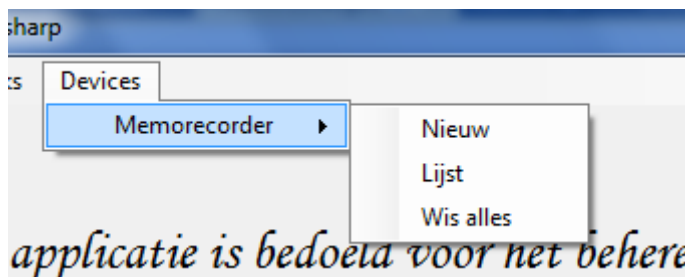
Naam	Type	Get	Set	Omschrijving
MaxCartridgeType	MemoCartridgeType	✓	✓	Cartridge met de grootste opslagcapaciteit die gebruikt kan worden bij deze memorecorder.

Stap 6: MemoRecorder Class testen

De MemoRecorder Class is nu compleet.

We gaan nu weer onze applicatie uitbreiden om de gemaakte classes te testen. Open daarom het project Tester in Visual Studio en open het formulier Main in design weergave.

Voeg aan de menubalk de optie **Devices** toe en voeg een submenu **Memorecorder** toe, met daarin de opties *Nieuw*, *Lijst* en *Wis alles* (zie Figuur 7 voor een voorbeeld).

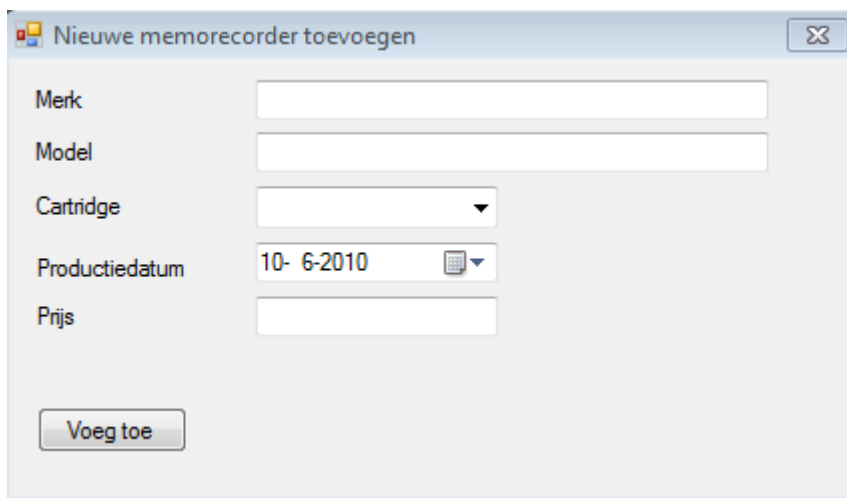


Figuur 7

Zodra de gebruiker klikt op Nieuw moet een formulier geopend worden waarmee een nieuwe memorecorder toegevoegd kan worden. Vanzelfsprekend moeten de ingevoerde memorecorders opgeslagen worden in het geheugen. Hiervoor definieer je in het formulier Main een static List van memorecorders:

```
public static List<MemoRecorder>
```

Maak vervolgens een nieuw formulier aan met de naam addMemoRecorder en plaats daarop controls voor het invoeren van een nieuwe memorecorder. Het ontwerp zou er als voorbeeld uit kunnen zien zoals weergegeven in Figuur 8.



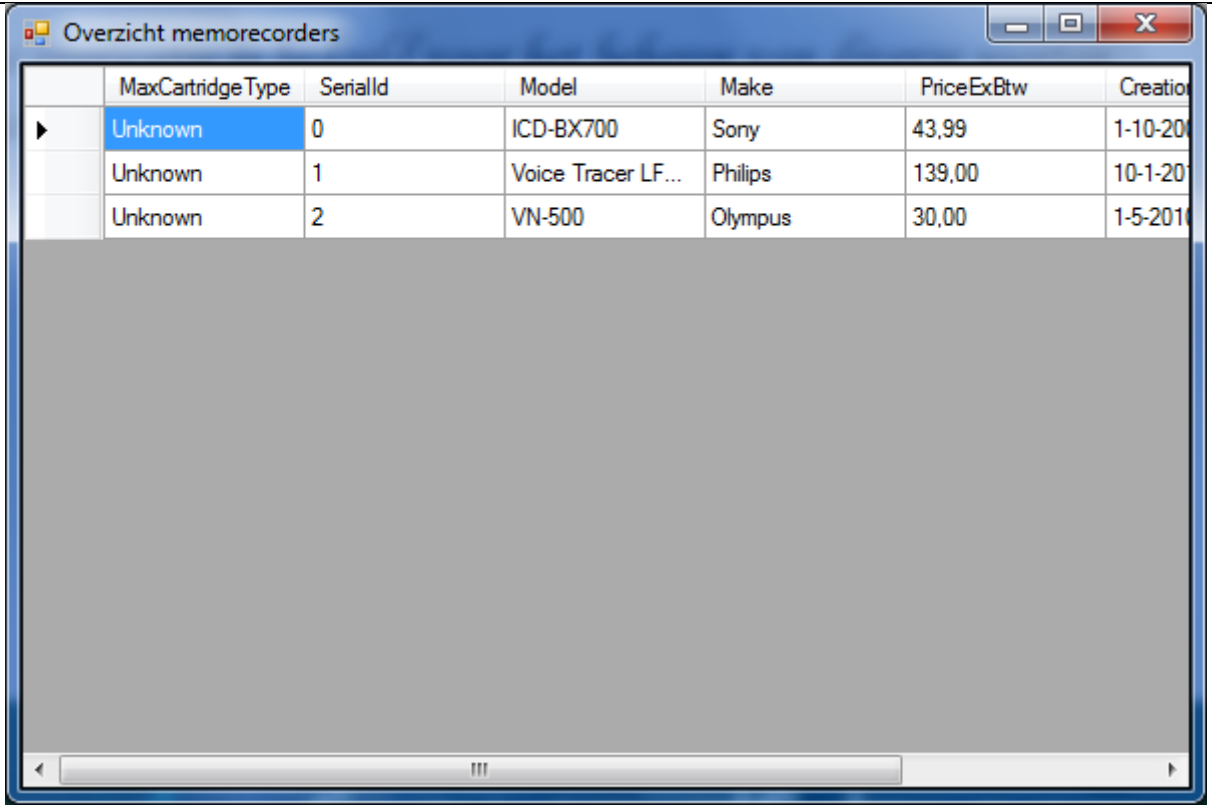
Figuur 8

Voor het invoeren van een cartridge type gebruiken we een combobox, die gevuld wordt met de waarden van de eerder gemaakte enumerator. De productiedatum wordt ingevoerd m.b.v. een datetempicker.

Door te klikken op de knop **Voeg toe** wordt de memorecorder toegevoegd aan de static list met memorecorders, waarna alle velden leeggemaakt worden en het veld Merk krijgt weer de focus, zodat de gebruiker direct verder kan met het invoeren van een volgende memorecorder.

We gaan de gebruiker ook nog de mogelijkheid geven de (in het geheugen) opgeslagen memorecorders te laten zien. Daarvoor maken we gebruik van een nieuw formulier met daarop een datagridview die gedocked wordt in het formulier (dock style = fill).

In runtime kan het formulier er uit komen te zien zoals weergegeven in Figuur 9.

A screenshot of a Windows application window titled "Overzicht memorecorders". The window contains a table with the following columns: MaxCartridgeType, SerialId, Model, Make, PriceExBtw, and Creation. The first three rows of data are visible, with the first row selected. The rest of the table area is greyed out.

	MaxCartridgeType	SerialId	Model	Make	PriceExBtw	Creation
▶	Unknown	0	ICD-BX700	Sony	43,99	1-10-2010
	Unknown	1	Voice Tracer LF...	Philips	139,00	10-1-2011
	Unknown	2	VN-500	Olympus	30,00	1-5-2011

Figuur 9

Tenslotte geven we de gebruiker via de menuoptie Devices->Memorecorder->Wis alles de mogelijkheid alle items uit de lijst te verwijderen. Zorg weer dat eerst om een bevestiging wordt gevraagd.

Opdracht 5, De *CdDiscMan* Class

De volgende Class die ontwikkeld wordt is de *CdDiscMan* Class. Deze Class stelt een CD Discman voor.

De *CdDiscMan* Class definiëren we als een concrete Class, d.w.z. dat we van deze Class objecten kunnen aan maken. De *CdDiscMan* Class is een derived Class van base Class *AudioDevice*.

Stap 1: Nieuwe Class aanmaken

Open in Visual Studio de *AudioDevices* solution. Selecteer in de solution explorer het *AudioDevices* project en vervolgens uit het context menu (rechtermuis klik) Add | New item... Kies in de template dialog de template Class en geef deze de naam *CdDiscMan.cs*.

Verander de namespace definitie in:

```
namespace AudioDevices.Devices
```

Verander de naam van de Class in *CdDiscMan* en zorg ervoor dat deze Class van *AudioDevice* is afgeleid.

Stap 2: Interface *IDisplay* toevoegen

De CD discmans van *SoundSharp* hebben een display om informatie te tonen. Aangezien er nog andere audiodevices zijn die ook displays hebben is besloten de functionaliteit die bij een display hoort samen te voegen in de interface *IDisplay*.

Selecteer in de solution explorer het *AudioDevices* project en vervolgens uit het context menu (rechtermuis klik) Add | New item... Kies in de template dialog de template Class en geef deze de naam *Interfaces.cs* in het geval je voor C# hebt gekozen of *Interfaces.vb* als je voor VB.NET hebt gekozen.

Verander de namespace definitie in:

```
namespace AudioDevices.Interfaces
```

Maak de interface *IDisplay* aan en definieer de volgende method in de interface:

```
string GetResolutionInfo();
```

Definieer de volgende properties in de *IDisplay* interface

```
int DisplayWidth {get; set;}  
int DisplayHeight {get; set;}  
int TotalPixels { get; }
```

Geef in de *CdDiscMan* Class aan dat deze Class de *IDisplay* interface implementeert.

Stap 3: Private fields toevoegen

Voeg in de CdDiscMan Class de volgende private fields toe:

Naam	Type	Omschrijving
mBSize*	int	Opslagcapaciteit van de CD Discman in MB.
displayWidth	int	Breedte van het display in pixels.
displayHeight	int	Hoogte van het display in pixels.
isEjected*	boolean	Geeft de ejection status van de CD Discman aan.

*mBSize

Definieer het private field mBSize als readonly en initialiseer deze met de waarde 700.

*isEjected

Initialiseer het private field isEjected met de waarde `false`.

Stap 4: Constructor toevoegen

Voeg aan de CdDiscMan Class de volgende constructor toe:

```
public CdDiscMan(int serialId)
```

Zorg ervoor dat het protected base field serialId geïnitieerd wordt met de waarde van de serialId parameter uit de constructor.

Stap 5: Methods toevoegen

De CdDiscMan Class moet een implementatie bevatten van de method die gedefinieerd is in de IDisplay interface.

```
GetResolutionInfo()
```

GetResolutionInfo() levert informatie op over de resolutie van het display van de CD Discman op in een string.

```
"Resolution: [totalpixels] pixels."
```

```
totalpixels = displayWidth * displayHeight.
```

```
DisplayStorageCapacity()
```

DisplayStorageCapacity() levert informatie op over de opslagcapaciteit van de CD Discman in een string.

```
"[capacity] mB.";
```

```
Eject()
```

De method Eject() toggled de isEjected boolean private field. Als isEjected **true** is wordt isEjected **false**. Als isEjected **false** is wordt isEjected **true**.

Stap 6: Properties toevoegen

Voeg in de CdDiscMan Class de volgende public properties toe. De kolommen Get en Set geven aan of de property respectievelijk een getter en/of een setter dient te bevatten.

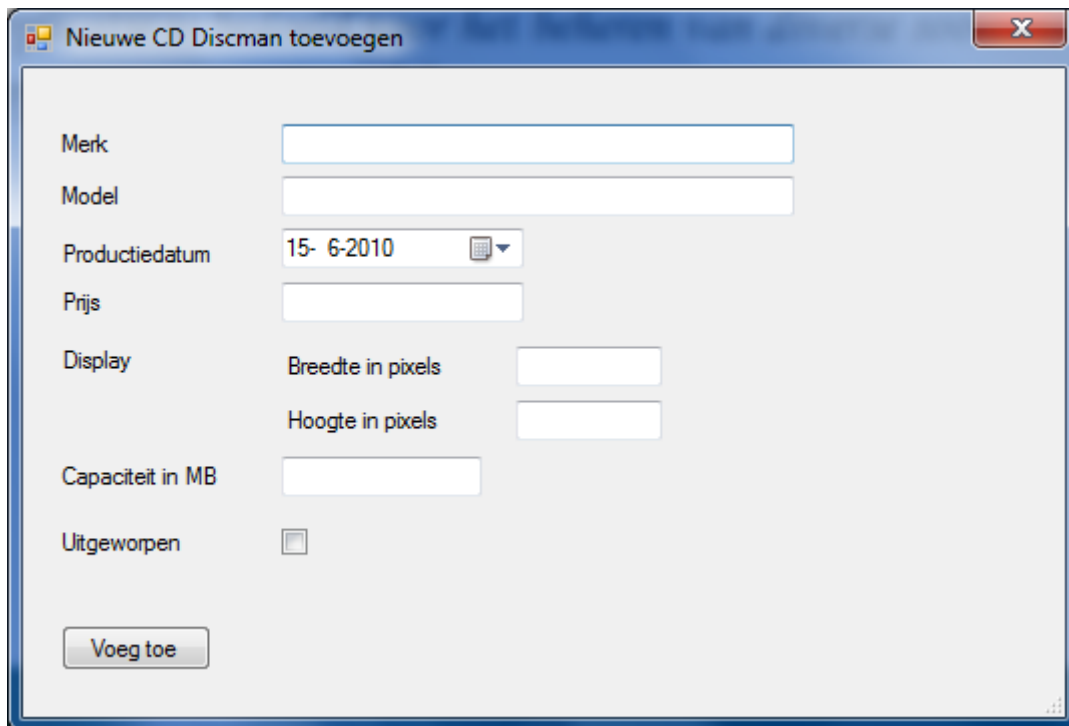
Naam	Type	Get	Set	Omschrijving
MbSize	int	✓	✗	De opslagcapaciteit van de CD Discman in MB.
DisplayWidth	int	✓	✓	Breedte van het display in pixels.
DisplayHeight	int	✓	✓	Hoogte van het display in pixels.
TotalPixels	Int	✓	✗	De resolutie van het display: displayWidth*displayHeight.
IsEjected	Boolean	✓	✗	De eject status van de CD Discman

Stap 7: CdDiscMan Class testen

Om onze nieuwe class te testen gaan we weer een optie toevoegen aan het menu van het formulier **Main**. Voeg onder devices de optie Discman toe met als opties:

- Nieuw
- Lijst
- Wis alles

Via Nieuw kan de gebruiker een nieuw formulier openen waarmee een nieuwe discman toegevoegd kan worden. De layout van het formulier kan er als volgt uit komen te zien (zie Figuur 10):



Figuur 10

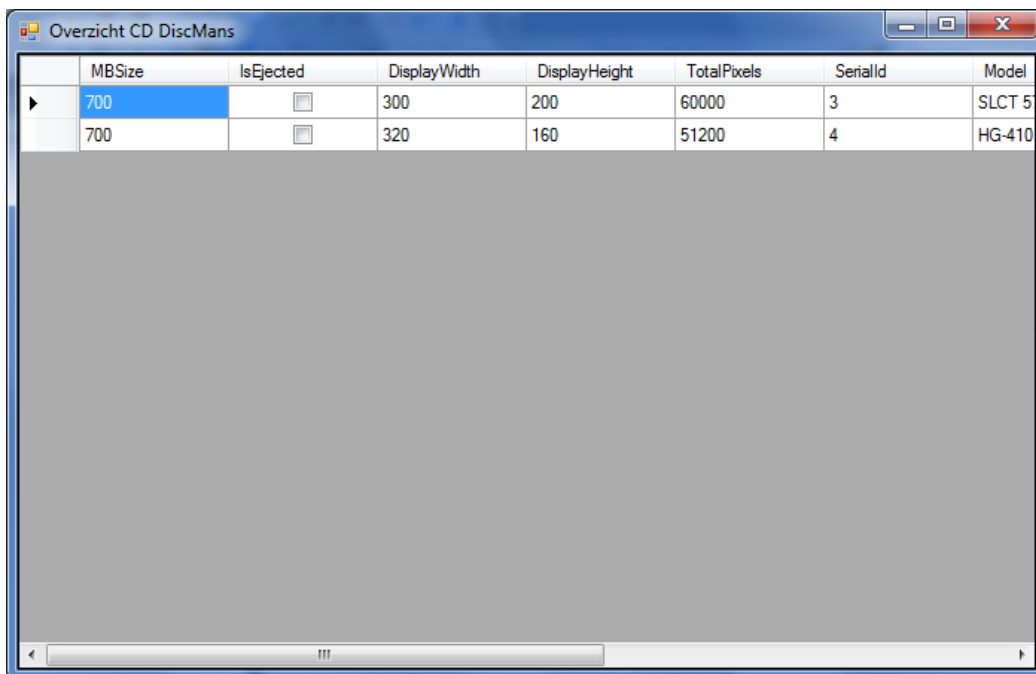
Ook nu moet het serienummer weer automatisch toegekend worden door het laatst gebruikte nummer met 1 te verhogen. De gebruiker moet dus niet zelf een ID invoeren.

Vanzelfsprekend moeten de ingevoerde discmans opgeslagen worden in het geheugen. Hiervoor definieer je in het formulier Main een static List van memorecorders:

```
public static List<CdDiscMan>
```

We gaan de gebruiker ook nog de mogelijkheid geven de (in het geheugen) opgeslagen discmans te laten zien. Daarvoor maken we gebruik van een nieuw formulier met daarop een datagridview die gedocked wordt in het formulier (dock style = fill).

In runtime kan het formulier er uit komen te zien zoals in .



Tenslotte geven we de gebruiker ook nog de mogelijkheid alle in het geheugen aanwezige discmans te verwijderen. Ook nu weer nadat eerst om een bevestiging is gevraagd.

Opdracht 6, De Mp3Player Class

De volgende Class die ontwikkeld wordt is de Mp3Player Class. Deze Class stelt een MP3 speler voor.

De Mp3Player Class definiëren we als een concrete Class, d.w.z. dat we van deze Class objecten kunnen aan maken. De Mp3Player Class is een derived Class van base Class AudioDevice.

Stap 1: Nieuwe Class aanmaken

Open in Visual Studio de AudioDevices solution. Selecteer in de solution explorer het AudioDevices project en vervolgens uit het context menu (rechtermuis klik) Add | New item... Kies in de template dialog de template Class en geef deze de naam Mp3Player.cs in het geval je voor C# hebt gekozen of Mp3Player.vb als je voor VB.NET hebt gekozen.

Verander de namespace definitie in:

```
namespace AudioDevices.Devices
```

Verander de naam van de Class in Mp3Player en zorg ervoor dat deze Class van AudioDevice is afgeleid.

Stap 2: Interface ITrackList toevoegen

Het is mogelijk een tracklist (zie opdracht 2) toe te kennen aan een mp3 speler. Alle functionaliteit die met TrackLists te maken hebben wordt opgenomen in de interface ITrackList.

Open de Interfaces.cs file (C#) of Interfaces.vb file (VB.NET) en voeg de nieuwe interface ITrackList toe.

Definieer de volgende methods in de ITrackList interface:

```
bool HasTracks();  
void AddTrackList(TrackList trackList);  
void RemoveTrackList();
```

Definieer de volgende property in de ITrackList interface:

```
TrackList TrackList {get;}
```

Geef in de Mp3Player Class aan dat deze Class de ITrackList interface implementeert. Geef tevens aan dat de Mp3Player Class de IDisplay interface implementeert.

Stap 3: Private fields toevoegen

Voeg in de Mp3Player Class de volgende private fields toe:

Naam	Type	Omschrijving
trackList	TrackList	De tracklist van de mp3 speler.
mBSize	int	Opslagcapaciteit van de mp3 speler in MB.
displayWidth	int	Breedte van het display in pixels.
displayHeight	int	Hoogte van het display in pixels.
picture	Image	Een afbeelding van de mp3 speler

`*mBSize`

Initialiseer het private field `mBSize` met de waarde 0.

Stap 4: Constructor toevoegen

Voeg aan de `Mp3Player` Class de volgende constructor toe:

```
public Mp3Player()
```

Ook nu moet het serienummer weer automatisch toegekend worden door het laatst gebruikte nummer met 1 te verhogen. De gebruiker moet dus niet zelf een ID invoeren.

Stap 5: Methods toevoegen

De `Mp3Player` Class moet een implementatie bevatten van de method gedefinieerd in de `IDisplay` interface.

```
GetResolutionInfo()
```

`GetResolutionInfo()` levert informatie op over de resolutie van het display van de mp3 speler in een string.

```
"Resolution: [totalpixels] pixels."
```

```
totalpixels = displayWidth * displayHeight.
```

De `Mp3Player` Class moet implementaties bevatten van de methods gedefinieerd in de `ITrackList` interface.

```
HasTracks()
```

`HasTracks()` geeft aan of de tracklist van de mp3 speler tracks bevat, dit wordt terug gegeven in een boolean.

```
AddTrackList(...)
```

```
public void AddTrackList(TrackList trackList)
```

De method `AddTrackList()` voegt een tracklist toe aan de mp3 speler. Als input parameter wordt een `TrackList` object meegegeven.

```
RemoveTrackList()
```

De method `RemoveTrackList()` verwijdert de tracklist van de mp3 speler.

Naast implementaties van de ITrackList en IDisplay methods heeft de Mp3Player Class ook een implementatie van de abstract method DisplayStorageCapacity().

```
DisplayStorageCapacity()
```

DisplayStorageCapacity() levert informatie op over de opslagcapaciteit van de mp3 speler in een string.

```
"[capacity] mB.";
```

Als de capacity niet bekend is (waarde 0 of minder) wordt de string

```
"Storage capacity unknown." teruggegeven.
```

Stap 6: Properties toevoegen

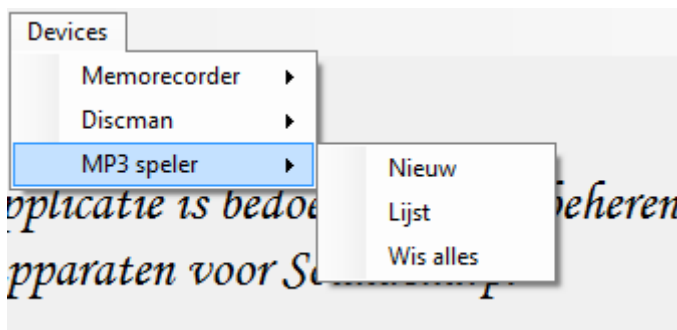
Voeg in de Mp3Player Class de volgende properties toe. De kolommen Get en Set geven aan of de property respectievelijk een getter en/of een setter dient te bevatten.

Naam	Type	Get	Set	Omschrijving
MbSize	int	✓	✓	De opslagcapaciteit van de mp3 speler in MB.
DisplayWidth	int	✓	✓	Breedte van het display in pixels.
DisplayHeight	int	✓	✓	Hoogte van het display in pixels.
TotalPixels	Int	✓	✗	De resolutie van het display: displayWidth*displayHeight.
TrackList	TrackList	✓	✗	De tracklist van de mp3 speler.
Picture	Image	✓	✓	Een afbeelding van de mp3 speler.

Stap 7: Mp3Player Class testen

De Mp3Player Class is nu complete en vanzelfsprekend gaan we onze testapplicatie zodanig aanpassen dat de gebruiker nieuwe mp3-spelers aan het assortiment kan toevoegen en ook een overzicht van de aanwezige spelers kan verkrijgen.

Breid eerst het menu Devices uit volgens Figuur 11.



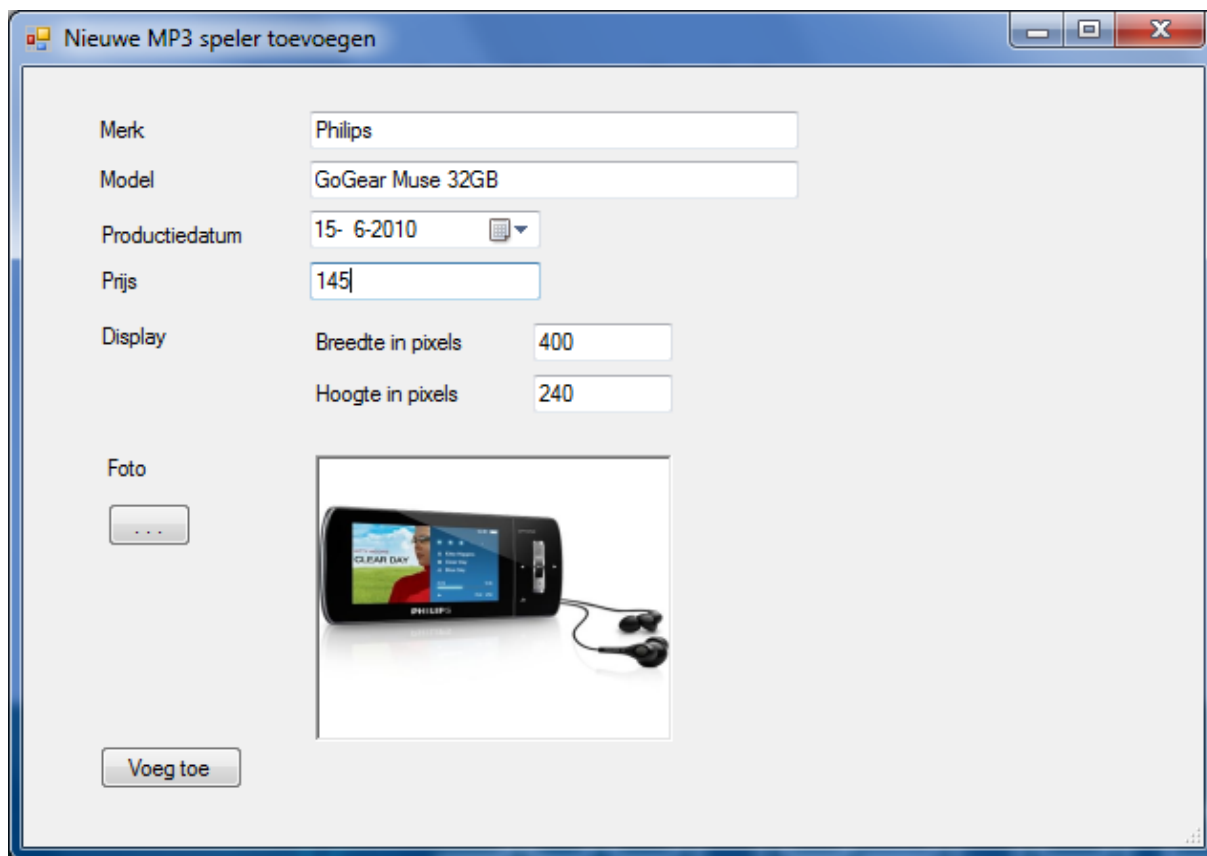
Figuur 11

De opties spreken inmiddels voor zich. Met de optie Nieuw kan een MP3-speler worden toegevoegd. Daarbij moet het mogelijk zijn een afbeelding van de harde schijf te selecteren met behulp van een

openFileDialog met als naam *ofdSelectImage*. Als standaardmap moet *Mijn afbeeldingen* worden geopend. Gebruik daarvoor de volgende code in het load event van het formulier:

```
ofdSelectImage.InitialDirectory =  
    Environment.GetFolderPath(Environment.SpecialFolder.MyPictures);
```

Een voorbeeld van het formulier zie je in Figuur 12.



Figuur 12

Voeg nu de testcode toe aan Tester om te controleren of de Mp3Player Class goed functioneert:

C#