Introductie SoundSharp

SoundSharp is een middelgrote handelsonderneming die in 1999 is opgericht door de huidige directrice Melanie Hendrikse. Vanaf het begin heeft SoundSharp zich gespecialiseerd in de verkoop van kwalitatief hoogstaande draagbare audioapparatuur. In het beginstadium werden voornamelijk walkmans, memorecorders en CD-Discmans van gerenommeerde merken verkocht. De laatste jaren wordt de markt voor draagbare audioapparatuur gedomineerd door mp3 spelers.

Management Team SoundSharp

Inmiddels bestaat SoundSharp uit 12 medewerkers met een gemiddelde leeftijd van 36 jaar. Binnen SoundSharp bestaan de afdelingen: Financiën, Inkoop/Verkoop en Technische support.

De dagelijkse gang van zaken wordt wekelijks besproken in het management team dat aangestuurd wordt door Melanie.

Medewerkers aan het woord

Willem van der Cuyk – Hoofd Inkoop/Verkoop

Hallo, ik ben Willem 47 jaar en sinds februari 2005 hoofd inkoop bij SoundSharp en lid van het Management Team. Ik ben eindverantwoordelijk voor de inkoop en verkoop bij SoundSharp. In mijn vak is het belangrijk nieuwe trends op de voet te volgen en goed op de hoogte te zijn van de markt.

Sinds een jaar of twee hebben we naast de reguliere draagbare audioapparatuur ook mp3 spelers in het assortiment. Kwaliteit staat hierbij bij SoundSharp zoals

altijd hoog in het vaandel. Over het algemeen kan je zeggen dat ons assortiment niet breed is maar wel kwalitatief hoogstaand. Veel van onze klanten zijn consumenten die graag wat extra willen betalen voor een goed product.

We produceren de audio apparatuur niet zelf, maar we zijn continu op zoek naar nieuwe modellen. Ik spendeer zelf veel tijd in het buitenland om bij de fabrikanten de nieuwste modellen te bewonderen en inkoopdeals te maken.

Sarissa de la Fuente – Medewerker afdeling financiën

Hoi, ik ben Sarissa 23 jaar en werk al vier jaar op de afdeling *financiën* bij SoundSharp. Ik zorg er samen met mijn collega voor dat alle rekeningen betaald worden en de facturen de deur uit gaan. Persoonlijk vind ik de sfeer bij SoundSharp altijd super, het is hier altijd leuk werken. Het voordeel van zo'n klein bedrijf is dat het lekker informeel is, als er iets is kan ik zo bij de directrice naar binnen kloppen.

Doordat we nog niet zo lang mp3 spelers verkopen is het voorraadbeheer hiervan nog niet geautomatiseerd. Nu doen we dat nog steeds in Excel en dat is niet handig, maar ik hoorde dat SoundSharp hier nog een systeem voor gaat ontwikkelen.

Léon Mourat - Supportmedewerker

Hoi, ik ben Léon en ben supportmedewerker van de Technische support afdeling van SoundSharp. Ik werk hier nog maar 8 maanden maar heb het erg naar mijn zin. In het kort komt het op neer dat kopers van audioapparatuur met technische problemen bij ons terecht kunnen voor eerstelijns support. Het contact is over het algemeen telefonisch of via de mail. Heel vaak kunnen we ze dan al meteen helpen. Als er echt reparaties uitgevoerd moeten worden doen we dat niet zelf maar laten we dat over aan de diverse support afdelingen van de fabrikanten. We zorgen dan voor de coördinatie en de communicatie met de klant.

Disclaimer

SoundSharp is een fictief bedrijf bedoeld voor educatieve doeleinden. Genoemde namen zijn tevens fictief. Overeenkomsten met bestaande bedrijven en personen berusten op louter toeval.

SoundSharp Object Oriënted module

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 2005 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:

C#

```
namespace AudioDevices.Tracks
```

VB.NET

```
Namespace 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:

C#:

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

VB.NET:

```
Public Sub New(ByVal seconds As Integer)
Public Sub New(ByVal minutes As Integer, ByVal seconds As Integer)
Public Sub New(ByVal hours As Integer, ByVal minutes As Integer,
ByVal seconds As Integer)
```

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

Voorbeeld:

C#

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

VB.NET

```
Dim t As Time = New Time(100)
```

Nu dient de structure t de volgende waarden te bevatten:

hours: 0

minutes: 1

seconds: 40

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:

C#

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

VB.NET

```
Public Sub New()
Public Sub New(ByVal id As Integer)
Public Sub New(ByVal id As Integer, ByVal name As String)
Public Sub New(ByVal id As Integer, ByVal artist As String, ByVal name As
String)
```

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	√	X	Naam van de uitvoerende artiest en naam van de track.
Length	time	X	√	Lengte van de track.
DisplayLength	string	√	X	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 Console Application binnen dezelfde Visual Studio 2005 solution.

Kies File | New Project… en maak een nieuwe Console 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.

C#

```
using AudioDevices.Tracks;
```

VB.NET

```
Imports Tracks
```

Voeg nu de volgende testcode toe aan Tester en controleer of de Track Class goed functioneert:

C#

```
Track t1 = new Track(1, "Nelly Furtado", "Maneater");
t1.AlbumSource = "Loose";
t1.Length = new Time(4, 31);
t1.Style = Category.Pop;

Console.WriteLine(t1.DisplayName);
Console.WriteLine(t1.DisplayLength);
Console.WriteLine("Tijd in seconden: {0}", t1.GetLengthInSeconds());
Console.WriteLine("Category: {0}", t1.Style.ToString());
Console.ReadLine();
```

VB.NET

```
Dim t1 As Track = New Track(1, "Nelly Furtado", "Maneater")
t1.AlbumSource = "Loose"
t1.Length = New Time(4, 31)
t1.Style = Category.Pop

Console.WriteLine(t1.DisplayName)
Console.WriteLine(t1.DisplayLength)
Console.WriteLine("Tijd in seconden: {0}", t1.GetLengthInSeconds())
Console.WriteLine("Category: {0}", t1.Style.ToString())
Console.ReadLine()

Console.ReadLine()
```

SoundSharp Object Oriënted module

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 2005 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:

C#

```
namespace AudioDevices.Tracks
```

VB.NET

```
Namespace 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

C#

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

VB.NET

```
Imports System.Collections.Generic
```

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

C#

```
private List<Track> tracks;
```

VB.NET

```
Private _tracks As List(Of Track)
```

Stap 3: Constructors toevoegen

Voeg aan de Track Class de volgende constructors toe:

C#

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

VB.NET

```
Public Sub New()
Public Sub New(ByVal tracks As List(Of Track))
```

Binnen de constructor zonder parameters dient het private field tracks met een lege instantie geïnitialiseerd 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(…)`

C#:
```
public void Add(Track t)
```

VB.NET:
```
Public Sub Add(ByVal t As Track)
```

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(…)`

C#:
```
public void Remove(Track t)
```

VB.NET:
```
Public Sub Remove(ByVal t As Track)
```

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:

C#:
```
List<Track>
```

VB.NET:
```
List(Of Track)
```

```
Shuffle()
```

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

C#: `List<Track>`

VB.NET: `List(Of 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	√	X	Aantal tracks in de tracklist
TotalTime*	time	√	X	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:

C#: `foreach (Track track in tracks)`

VB.NET `For Each track As Track In _tracks … Next`

M.b.v. de Track method GetLength() kan 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.

Voer nu in Tester de volgende testcode uit, om te controleren of de TrackList Class goed functioneert:

C#

```
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);

TrackList trackList = new TrackList();
trackList.Add(t1);
trackList.Add(t2);
trackList.Add(t3);
Console.WriteLine("Aantal tracks: {0}", trackList.Count);

trackList.Remove(t3);
Console.WriteLine("Aantal tracks: {0}", trackList.Count);

trackList.Add(t3);
Console.WriteLine("Totale tijd tracklist: {0}", trackList.TotalTime);

Console.WriteLine();
Console.WriteLine("Random lijst:");
List<Track> shuffled = trackList.Shuffle();
foreach (Track t in shuffled)
   Console.WriteLine(t.Id + " " + t.DisplayName);

Console.ReadLine();
```

VB.NET

```
Dim t1 As Track = New Track(1, "Prince", "Guitar")
t1.Length = New Time(4, 12)

Dim t2 As Track = New Track(2, "Nelly Furtado", "Say it Right")
t2.Length = New Time(4, 41)

Dim t3 As Track = New Track(3, "David Guetta & Chris Willis", "Love is
gone")
t3.Length = New Time(3, 50)

Dim trackList As TrackList = New TrackList()
trackList.Add(t1)
trackList.Add(t2)
trackList.Add(t3)
Console.WriteLine("Aantal tracks: {0}", trackList.Count)

trackList.Remove(t3)
Console.WriteLine("Aantal tracks: {0}", trackList.Count)

trackList.Add(t3)
Console.WriteLine("Totale tijd tracklist: {0}", trackList.TotalTime)

Console.WriteLine()
Console.WriteLine("Random lijst:")
Dim shuffled As List(Of Track) = trackList.Shuffle()
For Each t As Track In shuffled
   Console.WriteLine(t.Id.ToString() + " " + t.DisplayName)
Next
Console.ReadLine()
```

SoundSharp Object Oriënted module

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.

De AudioDevice Class definiëren we als een abstract 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 2005 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:

C#

```
namespace AudioDevices.Devices
```

VB.NET

```
Namespace Devices
```

Verander de naam van de Class in: AudioDevice en zorg ervoor dat je de Class abstract maakt. Gebruik hiervoor het keyword abstract (C#) of MustInherrit (VB.NET).

Stap 2: Protected fields toevoegen

Voeg in de AudioDevice Class de volgende protected fields toe:

Naam	Type	Omschrijving
serialId	integer	Unieke numerieke serialId van het device.
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:

C#

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

VB.NET

```
Protected Shared ReadOnly _btwPercentage As Double = 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:

C#

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

VB.NET

```
Dim diff As TimeSpan = DateTime.Now.Subtract(Me._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:

C#

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

VB.NET

```
Public MustOverride Function DisplayStorageCapacity() As String
```

Stap 4: Properties toevoegen

Voeg in de AudioDevice 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
SerialId	integer	√	√	Unieke numerieke serialId van het device.
Model	string	√	√	Modelnaam van het device.
Make	string	√	√	Merk van het device.
PriceExBtw	decimal	√	√	Prijs van het device exclusief BTW.
ConsumerPrice	decimal	√	X	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.

SoundSharp Object Oriënted module

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 2005 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:

C#

```
namespace AudioDevices.Devices
```

VB.NET

```
Namespace 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
maxCatridgeType	MemoCartridgetype*	Cartridge met de grootste opslagcapaciteit die gebruikt kan worden bij deze memorecorder.

MemoCatridgeType*

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
Unknown
```

Stap 3: Constructor toevoegen

Voeg aan de Class de volgende constructor toe:

C#

```
public MemoRecorder(int serialId)
```

VB.NET

```
Public Sub New(ByVal serialId As Integer)
```

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

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.

Voeg nu de testcode toe aanTester om te controleren of de MemoRecorder Class goed functioneert:

NB. Vergeet niet de devices namespace aan Tester toe te voegen:

C#

```
using AudioDevices.Devices;
```

VB.NET

```
Imports AudioDevices.Devices
```

C#

```
MemoRecorder memo = new MemoRecorder(1000);
memo.MaxCartridgeType = MemoCartridgeType.C90;
memo.Make = "Sony";
memo.Model = "FE190";
memo.PriceExBtw = 129.95M;
memo.CreationDate = DateTime.Now.AddMonths(-6);

Console.WriteLine(memo.DisplayIdentity(true, true));
Console.WriteLine(memo.DisplayStorageCapacity());
Console.WriteLine("Consumer price: {0:f2}", memo.ConsumerPrice);
Console.WriteLine(memo.GetDeviceLifeTime());
Console.ReadLine();
```

VB.NET

```
Dim memo As MemoRecorder = New MemoRecorder(1000)
memo.MaxCartridgeType = MemoCartridgeType.C90
memo.Make = "Sony"
memo.Model = "FE190"
memo.PriceExBtw = CType(129.95, Decimal)
memo.CreationDate = DateTime.Now.AddMonths(-6)

Console.WriteLine(memo.DisplayIdentity(True, True))
Console.WriteLine(memo.DisplayStorageCapacity())
Console.WriteLine("Consumer price: {0:f2}", memo.ConsumerPrice)
Console.WriteLine(memo.GetDeviceLifeTime())
Console.ReadLine()
```

SoundSharp Object Oriënted module

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 2005 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 in het geval je voor C# hebt gekozen of CdDiscMan.vb als je voor VB.NET hebt gekozen.

Verander de namespace definitie in:

C#

```
namespace AudioDevices.Devices
```

VB.NET

```
Namespace 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:

C#

```
namespace AudioDevices.Interfaces
```

VB.NET

```
Namespace Interfaces
```

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

C#

```
string GetResolutionInfo();
```

VB.NET

```
Function GetResolutionInfo() As String
```

Definieer de volgende properties in de IDisplay interface

C#

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

VB.NET

```
Property DisplayWidth() As Integer
Property DisplayHeight() As Integer
ReadOnly Property TotalPixels() As Integer
```

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:

C#

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

VB.NET

```
Public Sub New(ByVal serialId As Integer)
```

Zorg ervoor dat het protected base field serialId geinitialiseerd 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 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	√	X	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	√	X	De resolutie van het display: displayWidth*displayHeight.
IsEjected	Boolean	√	X	De eject status van de CD Discman

Stap 7: CdDiscMan Class testen

De CdDiscMan Class is nu compleet.

Voeg nu de testcode toe aan Tester om te controleren of de CdDiscMan Class goed functioneert.

NB. Vergeet niet de interfaces namespace aan tester toe te voegen:

C#

```
using AudioDevices.Interfaces;
```

VB.NET

```
Imports AudioDevices.Interfaces
```

C#

```
CdDiscMan discman = new CdDiscMan(1000);
discman.Make = "JVC";
discman.Model = "HG-410";
discman.PriceExBtw = 149.00M;
discman.DisplayWidth = 320;
discman.DisplayHeight = 160;
discman.CreationDate = DateTime.Parse("12-2-2006");

Console.WriteLine(discman.DisplayIdentity(true, true));
Console.WriteLine("Opslag capacity {0}",discman.DisplayStorageCapacity());
Console.WriteLine("Display resolution {0} pixels", discman.TotalPixels);
Console.WriteLine(discman.GetResolutionInfo());
Console.WriteLine("Consumer price: {0:f2}", discman.ConsumerPrice);
Console.WriteLine(discman.GetDeviceLifeTime());
Console.WriteLine("Eject status: {0}", discman.IsEjected);
discman.Eject();
Console.WriteLine("Eject status: {0}", discman.IsEjected);

Console.ReadLine();
```

VB.NET

```
Dim discman As CdDiscMan = New CdDiscMan(1000)
discman.Make = "JVC"
discman.Model = "HG-410"
discman.PriceExBtw = CType(149.0, Decimal)
discman.DisplayWidth = 320
discman.DisplayHeight = 160
discman.CreationDate = DateTime.Parse("12-2-2006")

Console.WriteLine(discman.DisplayIdentity(True, True))
Console.WriteLine("Opslag capacity {0}", discman.DisplayStorageCapacity())
Console.WriteLine("Display resolution {0} pixels", discman.TotalPixels)
Console.WriteLine(discman.GetResolutionInfo())
Console.WriteLine("Consumer price: {0:f2}", discman.ConsumerPrice)
Console.WriteLine(discman.GetDeviceLifeTime())
Console.WriteLine("Eject status: {0}", discman.IsEjected)
discman.Eject()
Console.WriteLine("Eject status: {0}", discman.IsEjected)
Console.ReadLine()
```

SoundSharp Object Oriënted module

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 2005 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:

C#

```
namespace AudioDevices.Devices
```

VB.NET

```
Namespace 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:

C#

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

VB.NET

```
Function HasTracks() As Boolean
Sub AddTrackList(ByVal trackList As TrackList)
Sub RemoveTrackList()
```

Definieer de volgende property in de ITrackList interface:

C#

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

VB.NET

```
ReadOnly Property TotalPixels() As Integer
```

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.

*mBSize

Initialiseer het private field mBSize met de waarde 0.

Stap 4: Constructor toevoegen

Voeg aan de Mp3Player Class de volgende constructor toe:

C#

```
public Mp3Player(int serialId)
```

VB.NET

```
Public Sub New(ByVal serialId As Integer)
```

Zorg ervoor dat het protected base field serialId geinitialiseerd wordt met de serialId parameter uit de constructor.

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(…)`

C#

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

VB.NET

```
Public Sub AddTrackList(ByVal trackList As TrackList) Implements
ITrackList.AddTrackList
```

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

`RemoveTrackList()`

De method RemoveTrackList() verwijderd 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	√	X	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	√	X	De resolutie van het display: displayWidth*displayHeight.
TrackList	TrackList	√	X	De tracklist van de mp3 speler.

Stap 7: Mp3Player Class testen

De Mp3Player Class is nu compleet.

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

C#

```
Mp3Player player = new Mp3Player(1000);
player.Make = "Creative";
player.Model = "Alpha";
player.PriceExBtw = 99.00M;
player.DisplayWidth = 120;
player.DisplayHeight = 80;
player.CreationDate = DateTime.Parse("1-1-2007");
player.MbSize = 1024;

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);

TrackList trackList = new TrackList();
trackList.Add(t1);
trackList.Add(t2);
trackList.Add(t3);

player.AddTrackList(trackList);

Console.WriteLine(player.DisplayIdentity(true, true));
Console.WriteLine("Capacity {0}",player.DisplayStorageCapacity());
Console.WriteLine("Display resolution {0} pixels", player.TotalPixels);
Console.WriteLine("Consumer price: {0:f2}", player.ConsumerPrice);
Console.WriteLine(player.GetDeviceLifeTime());

if (player.HasTracks())
{
   Console.WriteLine("The TrackList of this player has {0} tracks",
   player.TrackList.Count);

   foreach (Track t in player.TrackList.GetAllTracks())
      Console.WriteLine(t.DisplayName);

   player.RemoveTrackList();
   Console.WriteLine("The TrackList of this player has now {0} tracks",
   player.TrackList.Count);
 }
 Console.ReadLine();
```

VB.NET

```
Dim player As Mp3Player = New Mp3Player(1000)
player.Make = "Creative"
player.Model = "Alpha"
player.PriceExBtw = CType(99.0, Decimal)
player.DisplayWidth = 120
player.DisplayHeight = 80
player.CreationDate = DateTime.Parse("1-1-2007")
player.MbSize = 1024

Dim t1 As Track = New Track(1, "Prince", "Guitar")
t1.Length = New Time(4, 12)

Dim t2 As Track = New Track(2, "Nelly Furtado", "Say it Right")
t2.Length = New Time(4, 41)

Dim t3 As Track = New Track(3, "David Guetta & Chris Willis", "Love is
gone")
t3.Length = New Time(3, 50)

Dim trackList As TrackList = New TrackList()
trackList.Add(t1)
trackList.Add(t2)
trackList.Add(t3)

player.AddTrackList(trackList)

Console.WriteLine(player.DisplayIdentity(True, True))
Console.WriteLine("Capacity {0}", player.DisplayStorageCapacity())
Console.WriteLine("Display resolution {0} pixels", player.TotalPixels)
Console.WriteLine("Consumer price: {0:f2}", player.ConsumerPrice)
Console.WriteLine(player.GetDeviceLifeTime())

If (player.HasTracks()) Then
   Console.WriteLine("The TrackList of this player has {0} tracks",
   player.TrackList.Count)

   For Each t As Track In player.TrackList.GetAllTracks()
      Console.WriteLine(t.DisplayName)
   Next

   player.RemoveTrackList()
   Console.WriteLine("The TrackList of this player has now {0} tracks",
   player.TrackList.Count)
End If
Console.ReadLine()
```