Opdracht 1 – Betaalwijze

[Program.cs]

```
using System;

namespace Opdracht1
{
    class Program
    {
        static void Main(string[] args)
        {
            Program myProgram = new Program();
            myProgram.Start();
        }

        void Start()
        {
            PrintHeader("[CreditCard]");
            Payment ccPayment = new CreditCardPayment();
            ccPayment.Execute();

            PrintHeader("[PayPal]");
            Payment ppPayment = new PayPalPayment();
            ppPayment.Execute();

            PrintHeader("[PIN]");
            Payment pinPayment = new PINPayment();
            pinPayment.Execute();

            Console.ReadKey();
        }

        void PrintHeader(string header)
        {
            Console.ForegroundColor = ConsoleColor.Green;
            Console.WriteLine(header);
            Console.ResetColor();
        }
    }
}
```

[Payment.cs]

```
using System;

namespace Opdracht1
{
    public abstract class Payment
    {
        public void Execute()
        {
            EnterInformation();
            ExecutePayment();
            ConfirmPayment();
        }

        public void EnterInformation()
        {
            Console.WriteLine("entering information...");
        }

        public abstract void ExecutePayment();

        public void ConfirmPayment()
        {
            Console.WriteLine("sending confirmation mail...");
        }
    }
}
```

[CreditCardPayment.cs]

```
using System;

namespace Opdracht1
{
    public class CreditCardPayment : Payment
    {
        public override void ExecutePayment()
        {
            Console.WriteLine("processing CreditCard payment...");
        }
    }
}
```

[PayPalPayment.cs]

```
using System;

namespace Opdracht1
{
    public class PayPalPayment : Payment
    {
        public override void ExecutePayment()
        {
            Console.WriteLine("processing PayPal payment...");
        }
    }
}
```

[PINPayment.cs]

```
using System;

namespace Opdracht1
{
    public class PINPayment : Payment
    {
        public override void ExecutePayment()
        {
            Console.WriteLine("processing PIN payment...");
        }
    }
}
```

Opdracht 2 –Jukebox

[Program.cs]

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

namespace Opdracht2
{
    class Program
    {
        static void Main(string[] args)
        {
            Program myProgram = new Program();
            myProgram.Start();
        }

        void Start()
        {
            List<IVinylSingle> singles = ReadSingles("top2000-2017.csv");
            List<IVinylAlbum> albums = ReadAlbums("albums.csv");

            // create jukebox
            JukeBox jukeBox = new JukeBox(singles);

            // add albums
            foreach (IVinylAlbum album in albums)
            {
                jukeBox.Singles.Add(new AlbumAdapter(album));
            }

            // select single
            Console.Write("Select a single to play {0}..{1}: ", 1,
jukeBox.Singles.Count);
            int index = int.Parse(Console.ReadLine());

            while (index > 0)
            {
                jukeBox.SelectSingle(index);

                // play selected single
                jukeBox.Play();
                // jukeBox.Stoppen();

                Console.WriteLine();

                // select next single
                Console.Write("Select a number to play {0}..{1}: ", 1,
jukeBox.Singles.Count);
                index = int.Parse(Console.ReadLine());
            }

            Console.WriteLine("end of program...");
            Console.ReadKey();
        }
```

```
        List<IVinylSingle> ReadSingles(string filename)
        {
            List<IVinylSingle> singles = new List<IVinylSingle>();

            if (!File.Exists(filename))
                return singles;

            StreamReader reader = new StreamReader(filename);
            while (!reader.EndOfStream)
            {
                string line = reader.ReadLine();
                string[] items = line.Split(';');
                Single single = new Single(int.Parse(items[0]), items[1], items[2]);
                singles.Add(single);
            }
            reader.Close();

            return singles;
        }

        List<IVinylAlbum> ReadAlbums(string filename)
        {
            List<IVinylAlbum> albums = new List<IVinylAlbum>();

            if (!File.Exists(filename))
                return albums;

            StreamReader reader = new StreamReader(filename);
            while (!reader.EndOfStream)
            {
                string line = reader.ReadLine();
                string[] items = line.Split(';');
                Album album = new Album(items[0], items[1], int.Parse(items[2]));
                albums.Add(album);
            }
            reader.Close();

            return albums;
        }
    }
}
```

[IVinylSingle.cs]

```
namespace Opdracht2
{
    public interface IVinylSingle
    {
        void Play();
        void Stop();
        void Pause();
    }
}
```

[Single.cs]

```
using System;

namespace Opdracht2
{
    class Single : IVinylSingle
    {
        public int Ranking { get; private set; }
        public string Title { get; private set; }
        public string Artist { get; private set; }

        public Single(int ranking, string title, string artist)
        {
            this.Ranking = ranking;
            this.Title = title;
            this.Artist = artist;
        }

        public void Play()
        {
            Console.WriteLine("playing single '{0}'", ToString());
        }

        public void Stop()
        {
            Console.WriteLine("stopped single '{0}'", ToString());
        }

        public void Pause()
        {
            Console.WriteLine("paused single '{0}'", ToString());
        }

        public override string ToString()
        {
            return String.Format("{0}, {1} ({2})", Title, Artist, Ranking);
        }
    }
}
```

[IVinylAlbum.cs]

```
namespace Opdracht2
{
    public interface IVinylAlbum
    {
        void Play();
        void Stop();
        void Pause();
    }
}
```

[Album.cs]

```
using System;

namespace Opdracht2
{
    class Album : IVinylAlbum
    {
        public string Title { get; private set; }
        public string Band { get; private set; }
        public int Year { get; private set; }

        public Album(string title, string band, int year)
        {
            this.Title = title;
            this.Band = band;
            this.Year = year;
        }

        public void Play()
        {
            Console.WriteLine("playing album '{0}'", ToString());
        }

        public void Stop()
        {
            Console.WriteLine("stopped album '{0}'", ToString());
        }

        public void Pause()
        {
            Console.WriteLine("paused album '{0}'", ToString());
        }

        public override string ToString()
        {
            return String.Format("{0}, {1} ({2})", Title, Band, Year);
        }
    }
}
```

[AlbumAdapter.cs]

```
namespace Opdracht2
{
    public class AlbumAdapter : IVinylSingle
    {
        private IVinylAlbum album;

        public AlbumAdapter(IVinylAlbum album)
        {
            this.album = album;
        }

        public void Pause()
        {
            album.Pause();
        }

        public void Play()
        {
            album.Play();
        }

        public void Stop()
        {
            album.Stop();
        }
    }
}
```

[JukeBox.cs]

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

namespace Opdracht2
{
    public class JukeBox
    {
        public List<IVinylSingle> Singles { get; private set; }
        public IVinylSingle CurrentSingle { get; private set; }

        public JukeBox(List<IVinylSingle> singles)
        {
            this.Singles = singles;
            CurrentSingle = null;
        }

        public void SelectSingle(int index)
        {
            if ((index < 1) || (index > Singles.Count))
                throw new Exception(String.Format("Single {0} does not exist!",
index));

            CurrentSingle = Singles[index - 1];
        }

        public void Play()
        {
            if (CurrentSingle != null)
                CurrentSingle.Play();
        }

        public void Stop()
        {
            if (CurrentSingle != null)
                CurrentSingle.Stop();
        }
    }
}
```

Opdracht 3 – Kopieermachine

[Program.cs]

```
using System;

namespace Opdracht3
{
    class Program
    {
        static void Main(string[] args)
        {
            Program myProgram = new Program();
            myProgram.Start();
        }

        void Start()
        {
            CopyingMachine machine1 = CopyingMachine.GetUniqueInstance();
            CopyingMachine machine2 = CopyingMachine.GetUniqueInstance();

            Console.WriteLine("copying with 'machine 1'");
            machine1.Copy(10);
            machine1.Copy(23);

            Console.WriteLine();

            Console.WriteLine("copying with 'machine 2'");
            machine2.Copy(40);

            Console.ReadKey();
        }
    }
}
```

[CopyingMachine.cs]

```
using System;

namespace Opdracht3
{
    public class CopyingMachine
    {
        private int totalNumberOfCopies;
        private static CopyingMachine uniqueInstance;

        public int TotalNumberOfCopies
        {
            get { return totalNumberOfCopies; }
        }

        private CopyingMachine()
        {
            totalNumberOfCopies = 0;
        }

        public static CopyingMachine GetUniqueInstance()
        {
            if (uniqueInstance == null)
            {
                uniqueInstance = new CopyingMachine();
            }

            return uniqueInstance;
        }

        public void Copy(int nrOfCopies)
        {
            Console.WriteLine("copying, {0}x", nrOfCopies);
            totalNumberOfCopies += nrOfCopies;
            Console.WriteLine("total number of copies: {0}x", totalNumberOfCopies);
        }
    }
}
```

Opdracht 4 – Zonnepaneel systeem

[Program.cs]

```
using System;

namespace Opdracht4
{
    class Program
    {
        static void Main(string[] args)
        {
            Program myProgram = new Program();
            myProgram.Start();
        }

        void Start()
        {
            // create solarpanel system
            IObservable systeem = new SolarPanelSystem();
            IPanelController controller = new PanelController(systeem);

            // create a solarpanel display
            IObserver display = new SolarPanelDisplay(systeem);

            // perform a few measurements
            for (int i = 0; i < 10; i++)
                controller.NewMeasurement();

            Console.ReadKey();
        }
    }
}
```

[IPanelController.cs]

```
namespace Opdracht4
{
    interface IPanelController
    {
        void NewMeasurement();
    }
}
```

[PanelController.cs]

```
namespace Opdracht4
{
    class PanelController : IPanelController
    {
        private IObservable solarPanel;

        public PanelController(IObservable solarPanel)
        {
            this.solarPanel = solarPanel;
        }

        public void NewMeasurement()
        {
            solarPanel.NewMeasurement();
        }
    }
}
```

[IObserver.cs]

```
using System;

namespace Opdracht4
{
  public interface IObserver
  {
    void Update(int wattage);
  }
}
```

[SolarPanelDisplay.cs]

```
using System;

namespace Opdracht4
{
    public class SolarPanelDisplay : IObserver
    {
        private IObservable system;

        public SolarPanelDisplay(IObservable system)
        {
            this.system = system;
            system.AddObserver(this);
        }

        public void Update(int power)
        {
            Console.WriteLine("new measurement: {0} Watt", power);
        }
    }
}
```

[IObservable.cs]

```
namespace Opdracht4
{
    public interface IObservable
    {
        void AddObserver(IObserver observer);
        void RemoveObserver(IObserver observer);
        void NewMeasurement();
    }
}
```

[SolarPanelSystem.cs]

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

namespace Opdracht4
{
    public class SolarPanelSystem : IObservable
    {
        public int Power { get; set; }

        private List<IObserver> observers;
        private Random random;

        public SolarPanelSystem()
        {
            // create the list with observers
            observers = new List<IObserver>();

            random = new Random();
        }

        public void NewMeasurement()
        {
            // new measurement (between 300-400 Watt)
            Power = 300 + random.Next(100);

            NotifyObservers();
        }

        public void AddObserver(IObserver observer)
        {
            observers.Add(observer);
        }

        public void RemoveObserver(IObserver observer)
        {
            observers.Remove(observer);
        }

        private void NotifyObservers()
        {
            foreach (IObserver observer in observers)
                observer.Update(Power);
        }
    }
}
```