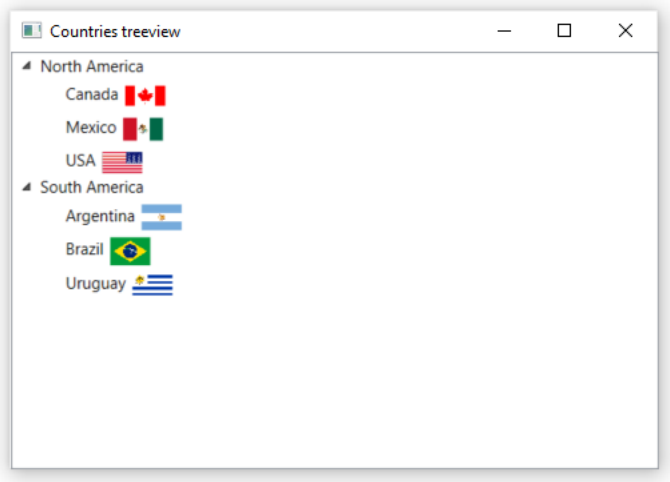
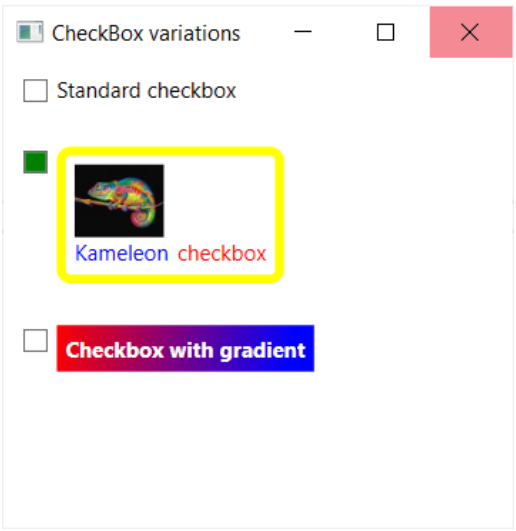
.NET Advanced Cheat Sheet

TreeView



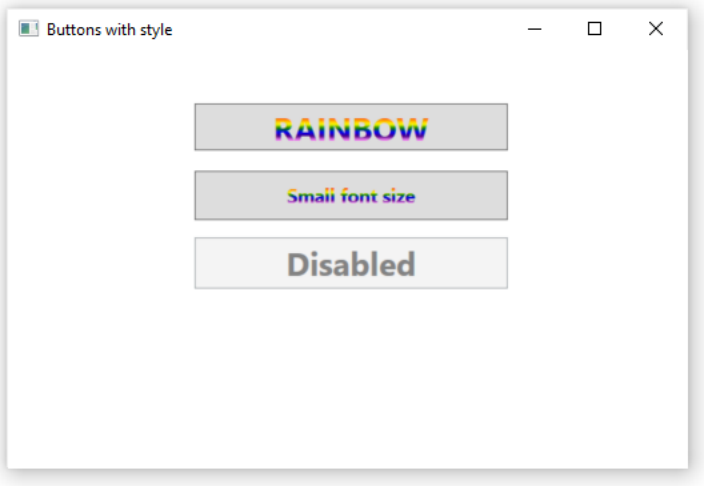

```
<TreeView>
  <TreeViewItem Header="North America" IsExpanded="True">
      <TreeViewItem>
          <TreeViewItem.Header>
              <StackPanel Orientation="Horizontal" Margin="3">
                  <TextBlock>canada</TextBlock>
                  <Image Source="images/canada.png" Width="25" Margin="3"></Image>
              </StackPanel>
          </TreeViewItem.Header>
        </TreeViewItem>
  </TreeViewItem>
</TreeView>
```

Special Checkoxes



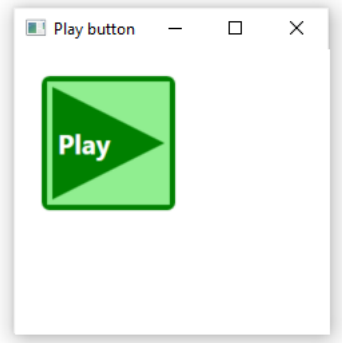

```
<Grid>
        <StackPanel>
            <CheckBox Grid.Row="0">Standard checkbox</CheckBox>
            <CheckBox Grid.Row="1" Background="Green">
                <Border Height="auto" Width="auto" Margin="5" Padding="2" BorderBrush="Yellow" BorderThickness="3, 3, 3, 3" CornerRadius="3
                    <StackPanel>
                        <Image Source="images\kameleon.jpg" HorizontalAlignment="Left" Height="20"></Image>
                        <StackPanel Orientation="Horizontal">
                            <TextBlock Text="Kameleon" FontSize="6" Foreground="Blue" Margin="0,0,4,0"></TextBlock>
                            <TextBlock Text="checkbox" FontSize="6" Foreground="Red"></TextBlock>
                        </StackPanel>
                    </StackPanel>
                </Border>
            </CheckBox>
            <CheckBox Grid.Row="2">
                <TextBlock Text="Checkbox with gradiant" FontWeight="Bold" Foreground="White" Padding="4,4,4,4">
                    <TextBlock.Background>
                        <LinearGradientBrush>
                            <LinearGradientBrush.GradientStops>
                                <GradientStop Color="Red"></GradientStop>
                                <GradientStop Color="blue" Offset="0.9"></GradientStop>
                            </LinearGradientBrush.GradientStops>
                        </LinearGradientBrush>
                    </TextBlock.Background>
                </TextBlock>
            </CheckBox>
        </StackPanel>
    </Grid>
```

Window Resource




```
<Window.Resources>
        <Style x:Key="myButtonStyle" TargetType="{x:Type Button}">
            <Setter Property="Foreground">
                <Setter.Value>
                    <LinearGradientBrush StartPoint="0,0" EndPoint="0,1">
                        <LinearGradientBrush.GradientStops>
                            <GradientStop Color="Red"></GradientStop>
                            <GradientStop Color="Orange" Offset="0.15"></GradientStop>
                            <GradientStop Color="Yellow" Offset="0.30"></GradientStop>
                            <GradientStop Color="Green" Offset="0.45"></GradientStop>
                            <GradientStop Color="Blue" Offset="0.60"></GradientStop>
                            <GradientStop Color="Indigo" Offset="0.75"></GradientStop>
                            <GradientStop Color="Violet" Offset="0.90"></GradientStop>
                        </LinearGradientBrush.GradientStops>
                    </LinearGradientBrush>
                </Setter.Value>
            </Setter>
            <Setter Property="FontWeight" Value="Bold"></Setter>
            <Setter Property="FontSize" Value="20"></Setter>
        </Style>
    </Window.Resources>
    <Grid>
        <StackPanel Width="230">
            <Button x:Name="buttonTop" Margin="0,10,0,10" Content="RAINBOW" Style="{StaticResource myButtonStyle}" Height="30"></Button>
            <Button x:Name="buttonMiddle" Margin="0,0,0,10" FontSize="15" Content="Small font size" Style="{StaticResource myButtonStyle}" 
            <Button x:Name="buttonBottom" Margin="0,0,0,0" Content="disabled" IsEnabled="False" Style="{StaticResource myButtonStyle}" Heig
        </StackPanel>
    </Grid>
```

Applicate resources



app.xaml

```
<Application.Resources>
        <ControlTemplate x:Key="myButtonTemplate" TargetType="{x:Type Button}">
            <Border CornerRadius="5" BorderBrush="Green" BorderThickness="3">
                <Grid Background="LightGreen">
                    <Polygon Points="0.0, 0.0, 1.0, 0.5, 0.0, 1.0" Stretch="Fill" Stroke="Green" Fill="Green" Margin="3"></Polygon>
                        <ContentPresenter HorizontalAlignment="Left" VerticalAlignment="Center" Margin="4,0,0,0"></ContentPresenter>
                </Grid>
            </Border>
        </ControlTemplate>
    </Application.Resources>
</Application>
```

Mainwindow.xaml

```
<Button Content="Play" FontWeight="Bold" Foreground="White" FontSize="20"
                HorizontalAlignment="Left" VerticalAlignment="Top" Margin="20,20,0,0"
                Width="100" Height="100" Template="{StaticResource myButtonTemplate}"/>
```

DataTemplate

```
<ComboBox x:Name="gamesCombobox" SelectionChanged="gamesCombobox_SelectionChanged" HorizontalAlignment="Left" Margin="190,20,0,0" VerticalA
    <ComboBox.ItemTemplate>
        <DataTemplate>
            <StackPanel Orientation="Horizontal" Margin="2">
                <TextBlock Text="{Binding GameId}" />
                <TextBlock Text=" - " />
                <TextBlock Text="{Binding Name}" />
            </StackPanel>
        </DataTemplate>
    </ComboBox.ItemTemplate>
</ComboBox>
```

INotifyPropertyChanged

1. Implement Interface

```
public class Game : INotifyPropertyChanged
```

2. Maak private field voor property die notify gaat implementeren (in dit geval _rating)

```
private double _rating;
```

3. Maak event aan en OnPropertyChanged functie aan

```
public event PropertyChangedEventHandler PropertyChanged;
private void OnPropertyChanged([CallerMemberName] string caller = "")
{
    if (PropertyChanged != null)
    {
        PropertyChanged(this, new PropertyChangedEventArgs(caller));
    }
}
```

4. Roep bij de setter van de property de functie op (Je kan ook if check doen om te controleren of de value echt anders is dan de current, zo niet moet je de functie niet oproepen)

```
public double Rating
{
    get => _rating;
    set
    {
        _rating = value;
        OnPropertyChanged(nameof(Rating));
    }
}
```

Converter

1. Implement Interface

```
public class RatingConverter : IValueConverter
```

2. Convert = Van Source (c#) naar Target (xaml) gaan

```
public object Convert(object value, Type targetType, object parameter, CultureInfo culture)
{
    double rating = System.Convert.ToDouble(value);
    return rating * 10;
}
```

3. ConvertBack = Van Target (xaml) naar Source (c#)

```
public object ConvertBack(object value, Type targetType, object parameter, CultureInfo culture)
{
    double multipliedRating = System.Convert.ToDouble(value);
    return multipliedRating / 10;
}
```

ObservableCollection

```
private readonly ObservableCollection<Employee> _employees;
public ReadOnlyObservableCollection<IOrder> Orders { get; }

// In de Constructor
_orders = new ObservableCollection<IOrder>();
Orders = new ReadOnlyObservableCollection<IOrder>(_orders);
```

Extension

Een extension is altijd STATIC, FUNCTIES OOK!!!!!!!!!!!!

Functie parameter begint altijd met THIS

```
public static class IntegerExtensions
{
    public static int CircularIncrement(this int value, int minimum, int maximum)
    {
        if (value + 1 > maximum || value + 1 < minimum)
```

```
        {
            return minimum;
        }
        return value + 1;
    }
```

Queue

1. Maak een property voor de Queue

```
private Queue<IOrder> ordersInQue = null;
```

2. Initializeer de Queue in de constructor

```
public Chef(FrontDesk frontDesk, IChefActions chefActions)
        {
            ordersInQue = new Queue<IOrder>();
            frontDesk.OrderCreated += FrontDesk_OrderCreated;
            chef = chefActions;
        }
```

3. Enqueue de nodige waarden/lijst

```
private void FrontDesk_OrderCreated(object sender, OrderEventArgs args)
        {
           ordersInQue.Enqueue(args.Order);
        }
```

4. Maak uw Queue

1ste manier

```
//TODO: check if the queue contains an order. If so -> process it.
while (ordersInQue.Count != 0)
      {
          Order order = (Order)ordersInQue.Dequeue();
          order.IsStarted = true;
          for (int i = 0; i < order.NumberOfBurgers; i++)
          {
              chef.CookBurger();
          }
          order.IsCompleted = true;
          chef.TakeABreather();
      }
  }
```

2de manier

```
//TODO: check if the queue contains an order. If so -> process it.
if (queue.Count > 0)
{
    foreach (Order order in queue)
    {
        order.IsStarted = true;
        for (int i = 0; i < order.NumberOfBurgers; i++)
        {
            chefActions.CookBurger();
        }
        order.IsCompleted = true;
        chefActions.TakeABreather();
    }
    queue.Clear();
}
```

Delegate

Maak een delegate method

```
public delegate void OrderCreatedHandler(object sender, OrderEventArgs args);
```

Maak een event die EventArgs implenteerd

```
public class OrderEventArgs : EventArgs
{
    public IOrder Order { get; }

    public OrderEventArgs(IOrder order)
    {
        Order = order;
    }
}
```

Roep delegate method met als een event

```
Class Frontdesk

public void AddOrder(int numberOfBurgers)
{
    _order.Add(new Order(numberOfBurgers));
    for (int i = 0; i < _order.Count; i++)
    {
        OrderEventArgs orderEventArgs = new OrderEventArgs(_order[i]);
        OrderCreated(this, orderEventArgs);
    }
}

public event OrderAggregate.OrderCreatedHandler OrderCreated;
```

Wanneer je de methode wilt gebruiken doe een `'+='` dan komt alles van zelf hoop ik voor u.

Paars is wat er komt.

```
Class Chef

public Chef(FrontDesk frontDesk, IChefActions chefActions)
{
    ordersInQue = new Queue<IOrder>();
    frontDesk.OrderCreated += FrontDesk_OrderCreated;
    chef = chefActions;
}
private void FrontDesk_OrderCreated(object sender, OrderEventArgs args)
{
   ordersInQue.Enqueue(args.Order);
}
```

Unit Testing

1. Voeg TextFixture boven de classe

```
[TestFixture]
public class RomanNumberConverterTests {
  private RomanNumberConverter _romanNumberConverter;
```

2. Om iets voor een test te initalizeren gebruiken we SetUp

```
[SetUp]
public void SetUp()
{
    _romanNumberConverter = new RomanNumberConverter();
}
```

3. Een Test die controleert of de correcte exception gethrowed word

```
[Test]
public void Convert_ShouldThrowArgumentExceptionWhenValueIsNotAString()
{
    var obj = new object();

    Assert.That(() => _romanNumberConverter.Convert(obj, null, null, null),
        Throws.InstanceOf<ArgumentException>()
        .With.Message.Contains("string"));
}
```

4. Een Test met meerdere USECASES

```
[Test]
[TestCase("")]
[TestCase("a")]
[TestCase("123a")]
public void Convert_ShouldReturnInvalidNumberWhenTheValueCannotBeParsedAsAnInteger(string value)
{
    string converted = _romanNumberConverter.Convert(value, null, null, null).ToString();

    Assert.That(converted, Is.EqualTo("Invalid number"));
}
```

WIRING

1. Ga naar Properties → AssemblyInfo.cs file van de Layer die je wil gebruiken in de presentatielaag.

```
using System.Runtime.CompilerServices;
[assembly: InternalsVisibleTo("ShoppingListApp.Presentation")]
```

2. Ga naar App.xaml.cs file

```
// Vergeet Usings niet

public partial class App : Application
{
    protected override void OnStartup(StartupEventArgs e)
    {
        ShoppingListContext context = new ShoppingListContext();
        context.CreateOrUpdateDataBase();
        //TODO: create an instance of a class that implements IShoppingListRepository
        ShoppingListDbRepository repository = new ShoppingListDbRepository(context);
        //TODO: create an instance of MainWindow
        MainWindow window = new MainWindow(repository);
        //TODO: show the instance of MainWindow
        window.Show();
    }
}
```

LINQ

Where

```
public IList<Person> FilterOutCatLoversAndDogLovers(List<Person> persons)
{
    //Tip: use the "ToList" extension method to convert an IEnumerable to a List
    var result = from person in persons
                 where person.FavoriteAnimal.ToLower() != "cat" && person.FavoriteAnimal.ToLower() != "dog"
                 select person;
    return result.ToList();
}
```

Select

```
public IList<PersonSummary> ConvertPersonsToPersonSummaries(IEnumerable<Person> persons)
{
    var result = from person in persons
                 select new PersonSummary
                 {
                     Id = person.Id,
                     FullName = $"{person.Firstname} {person.Lastname}",
                     IsChild = person.Age < 18
                 };

    return result.ToList();
}
```

OrderBy

```
public string[] SortWordsByLengthDescending(string[] words)
{
    //Tip: use the "ToArray" extension method to convert an IEnumerable to an Array
    var results = from word in words
                  orderby word.Length descending
                  select word;
    return results.ToArray();
}

public IList<Person> SortPersonsOnAscendingAgeAndThenOnFavoriteAnimalDescending(List<Person> persons)
{
    //Tip: use the "ToList" extension method to convert an IEnumerable to a List
    var results = from person in persons
                  orderby person.Age, person.FavoriteAnimal descending
                  select person;
    return results.ToList();
}
```

Join

```
public int[] GetIntersection(int[] firstList, int[] secondList)
{
    //Tip: use the "ToArray" extension method to convert an IEnumerable to an Array
    var result = from number in firstList
                 join number2 in secondList on number equals number2
                 select number2;
    return result.ToArray();
}

public IList<string> Merge2GroupsIntoDuosByAgeUsingJoin(List<Person> group1, List<Person> group2)
{
    //Tip: use the "ToList" extension method to convert an IEnumerable to a List
    var result = from person in group1
                 join person2 in group2 on person.Age equals person2.Age
```

```
                select $"{person.Firstname} and {person2.Firstname}";
    return result.ToList();
}
```

GroupBy

```
//Tip: use the "ToList" extension method to convert an IEnumerable to a List
public IList<IGrouping<bool, int>> GroupNegativeAndPositiveNumbers(int[] numbers)
{
    //Tip: Zero can be interpreted as a positive number
    return numbers.GroupBy(x => x >= 0).ToList();
}

public IList<AnimalLoverCollection> GroupPersonsByFavoriteAnimal(List<Person> persons)
{
    var result = from person in persons
                 group person by person.FavoriteAnimal into animalCollection
                 select new AnimalLoverCollection
                 {
                     Animal = animalCollection.First().FavoriteAnimal,
                     Persons = animalCollection
                 };
    return result.ToList();
}
```

Disconnected scenario

Door ‘using’ maak een new context. Alles wat je in using doet is in een disconnected scenario

```
var customer = _bankContext.Customers.Find(newAccount.CustomerId);
customer.Accounts.Add(newAccount);

using (var newContext = new BankContext())
{
    newContext.SaveChanges();
    CommitChanges();
}

public void CommitChanges()
{
    _bankContext.SaveChanges();
}
```

Struct

In een struct is er geen ‘NULL’

```
struct Coordinate
{
    public int x;
    public int y;
}

Coordinate point = new Coordinate();
Console.WriteLine(point.x); //output: 0
Console.WriteLine(point.y); //output: 0
```

```
public struct OrderNumber
{
    private const int MINIMUM = 1;
    private const int MAXIMUM = 99;
    private static int _sequence;
    public int Sequence { get; }
```

```
    public OrderNumber(int orderNumber)
    {
        Sequence = orderNumber;
    }
    public static OrderNumber CreateNext()
    {
        _sequence = IntegerExtensions.CircularIncrement(_sequence, MINIMUM, MAXIMUM);
        return new OrderNumber(_sequence);
    }
    public override string ToString()
    {
        return "#" + Sequence;
    }
}
```