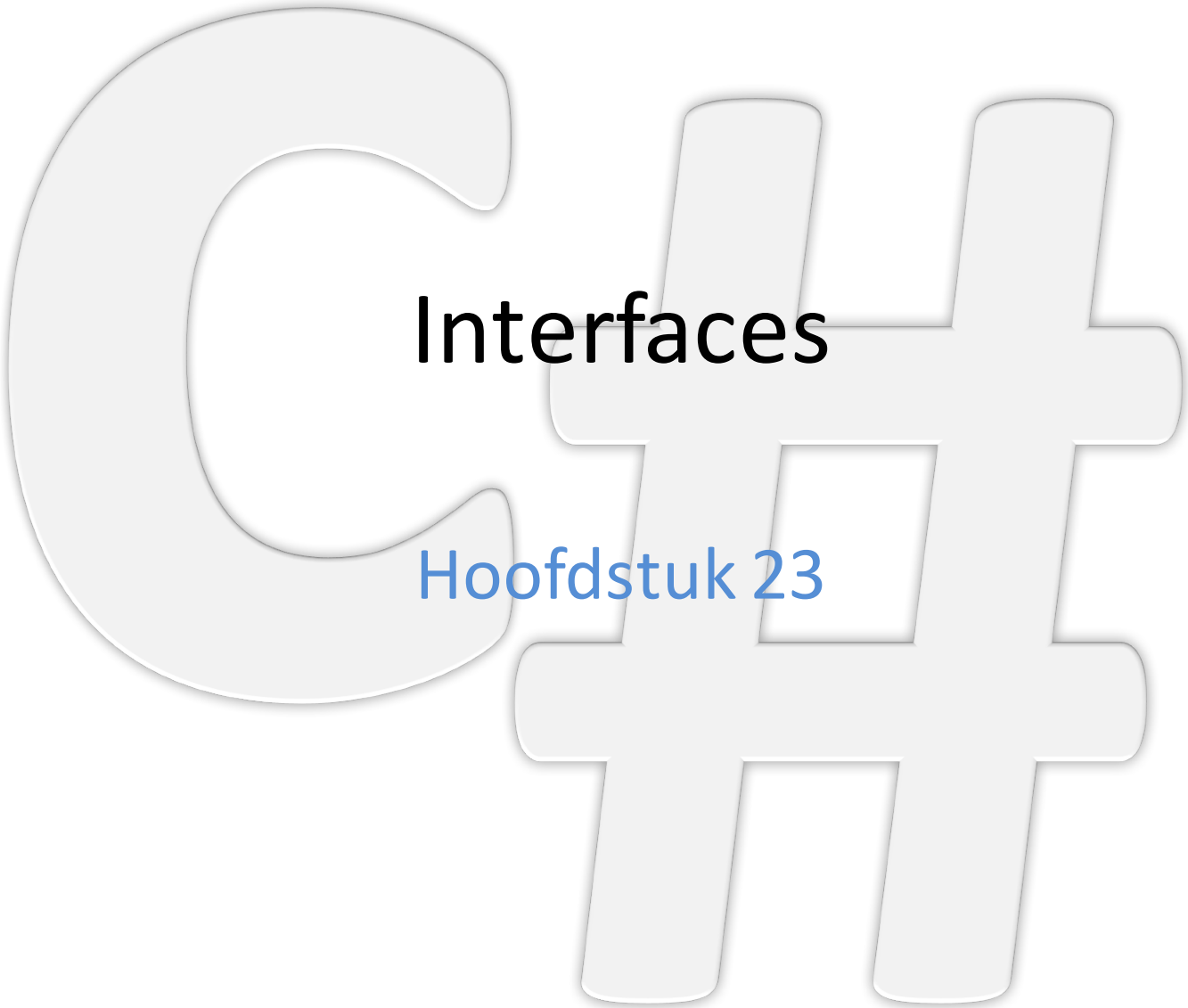


Interfaces

Hoofdstuk 23

Gradaties overerving

Klassieke overerving

- Iets functioneels uitbreiden
  - Code duplicatie
  - Niet teveel aanpassingen aan bestaande code
- Objecten van sub- en superklassen
- Bv. Ticket en VIP Ticket

Abstracte klassen

- Iets deels functioneels uitbreiden
  - Zelfde nut als overerving
  - Aanroep vastleggen voor subklassen
- Geen objecten van superklasse
- Bv. Vorm en Rechthoek

Interfaces

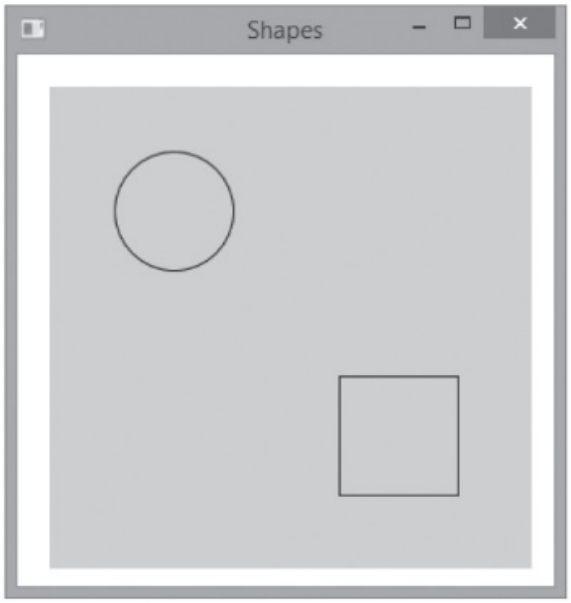
- Structuur vastleggen van klasse
  - Samenwerking programmeurs
  - Lijsten opstellen met objecten vanuit zelfde interface
  - Meervoudige overerving
- Geen objecten van interface, alles abstract
- Bv. IGame, IBalloon

Herhaling syntax klassieke overerving

```
public class Vierkant
{
    protected int zijde;

    public Vierkant(int z)
    {
        this.zijde = z;
    }

    public virtual int Oppervlakte()
    {
        return zijde * zijde;
    }
}
```

```
public class Kubus
    : Vierkant
{
    public Kubus(int z)
        : base(z)
    { }

    public override int Oppervlakte()
    {
        return base.Oppervlakte() * 6;
    }
}
```

Resultaat



Figuur 24.1 Het beeldscherm van de vormen (shapes) met gebruik van polymorfie

Herhaling syntax: abstracte klasse

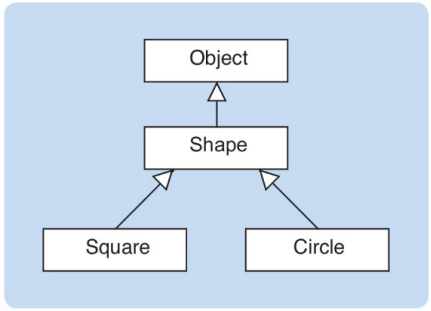
```
public abstract class Shape
{
    protected int xCoord, yCoord;
    protected int size;
    protected Pen pen = new Pen(Color.Black);

    public void MoveRight()
    {
        xCoord += 10;
    }

    public abstract void Display(Graphics drawArea);
}
```

Van deze klasse kan je **geen** objecten maken.
Logisch, want een shape heeft fysisch nog geen betekenis.

Geen constructor

Van deze methode bestaat (nog) geen implementatie.
Deze moet ingevuld worden in de subklassen.
Logisch, want een Shape is pas toonbaar als we al zijn eigenschappen bepaald hebben.

```
public class Circle : Shape
{
    public override void Display(Graphics drawArea)
    {
        drawArea.DrawEllipse(pen, xCoord, yCoord, size, size);
    }
}
```



Figuur 24.2 Klassediagram voor de vormklassen

Vraagjes

- Kan er ook staan? `public class Kubus: Vierkant,Shape`
- Waarvoor staat je base(z) in constructor van Kubus?
- Kan je base(z) weglaten in constructor van Kubus? Indien niet, wat moet er gebeuren opdat je het toch kan weglaten?
- Kan je virtual weglaten in klasse Vierkant? Indien niet, wanneer kan de het wel?
- Moet override daar staan in klasse Kubus?
- Waarvoor dient base.Oppervlakte()? Hoe kan je dat weglaten?
- Stel je hebt nieuwe klasse Form1
  - Declareer en initialiseer nu een instantie variabele van de klasse Kubus en de klasse Vierkant?
  - ToString:
    - Schrijf functie ToString die oppervlakte weergeeft van objecten Kubus en Vierkant?
    - Waar maak je deze functie aan?
    - Gebruik ToString in een MessageBox binnen Form1?
  - Hoe kan je binnen Form1 zijde aanpassen van Vierkant en Kubus object?

Vraagjes

- Waarvoor staat abstract class (waarom,wanneer)? Als je abstract weglaat, wat moet je dan nog aanpassen?
- Stel je hebt nieuwe klasse MainWindow
  - Declareer en initialiseer nu een instantie variabele van de klasse Shape en de klasse Circle?
  - Kan je de functie MoveRight laten inwerken op beide objecten?
  - Zal de functie `drawArea.DrawEllipse(pen, xCoord, yCoord, size, size);` problemen opleveren wanneer je ze laat uitvoeren op een Circle object?

In dit hoofdstuk …

- Interfaces gebruiken voor de structuur van een programma
  - Onderlinge aansluiting, meerdere programmeurs
  - Polymorfie: lijst in 1 keer itereren
- Interfaces versus abstracte klassen

Inleiding

- Een interface beschrijft de uiterlijke verschijningsvorm van een klasse
  - *Geen implementaties van methodes of properties*
- Niet te verwarren met grafische user interface
- Doel:
  - Ondersteunen van het ontwerp
  - Bevorderen van onderlinge aansluiting (eng: cohesion)
- Het concept interface bestaat eveneens in Java

Syntax interface

```
public interface IBalloon
{
    void ChangeSize(int diameter);
    void Display(Graphics drawArea);
    int X { get; set; }
}
```

- Geen `class` sleutelwoord
- Geen access modifiers bij methoden en properties
- Dit is puur een beschrijving van diensten: uiterlijke verschijningsvorm van een klasse
- Het bevat *geen* implementatie!
- `get`: mogelijkheid tot lezen, `set`: mogelijkheid tot schrijven
  → dit zijn ook *geen* implementaties!

Interface

- Kan gecompileerd worden samen met andere klassen
  - Wordt gecheckt of klasse op juiste manier wordt gebruikt
- Kan niet uitgevoerd worden

```
public interface IBalloon
{
    void ChangeSize(int diameter);
    void Display(Graphics drawArea);
    int X { get; set; }
}
```

```
public class Balloon : IBalloon
{
    private int diameter;
    private int x, y;

    public void ChangeSize(int diameter)
    {
        this.diameter = diameter;
    }

    public int X
    {
        get { return x; }
        set { x = value; }
    }

    public void Display(Graphics drawArea)
    {
        Pen pen = new Pen(Color.Black);
        drawArea.DrawEllipse(pen, x, y, diameter, diameter);
    }
}
```

Een klasse kan één of meerdere interfaces implementeren

De klasse is **verplicht** om alle methodes en properties uit de interface te implementeren

Interfaces van interface

- Interfaces kunnen ook overerven van andere interfaces

```
public interface IColoredBalloon : IBalloon
{
    void SetColor(Color c);
}
```

Nut 1: Aansluiting

- Interfaces worden vaak eerst uitgeschreven aan de hand van een analyse
- Vervolgens worden de interfaces verdeeld onder de programmeurs en geïmplementeerd
- Voorbeeld:

Stel Jan moet interface `IBalloon` implementeren en Mieke moet `IGame` implementeren. Bovendien heeft Mieke de implementatie van `IBalloon` nodig. Omdat je werkt met interfaces hoeft Mieke niet te wachten en kan ze onmiddellijk starten met de implementatie van `IGame`, eventueel gebruik makend van een "Mock" implementatie van `IBalloon`.
Een "Mock" implementatie is een zeer eenvoudige implementatie die weinig of geen functionaliteit bevat, maar louter als bedoeling heeft testen of verdere implementaties te bewerkstelligen.

Demo Interface

"Spel" waarbij voorwerp random horizontaal over scherm verplaatst bij elke drukknop.

Stel 3 programmeurs: Interface, Voorwerp, Spel

- Interface programmeur moet weten wat spel doet en hoe voorwerp eruit ziet
- Spel programmeur moet weten wat voorwerp doet

Brainstorm meeting om interfaces vast te leggen van Spel en Voorwerp.

- IShape -> Balloon
  - Display op graphics, X
- IGame -> Game
  - Replace van IShape op graphics
- GUI
  - Game tonen met een IShape

Onderlinge aansluiting

- Het afspreken van standaarden zorgt ervoor dat onderlinge uitwisselbaarheid verbetert, bijvoorbeeld:
  - Stopcontacten
  - Audio- en videoaansluitingen
  - DVD standaarden
  - …
- Het concept interface brengt deze voordelen naar de softwarewereld

Nut 2: Gezamelijke behandeling

```
public interface IDisplayable
{
    void Display(Graphics drawArea);
}
```

Alle klassen die moeten getekend kunnen worden moeten deze interface implementeren

```
public class Square : IDisplayable
{
    private int x, y;
    private int size;

    public void Display(Graphics drawArea)
    {
        Pen pen = new Pen(Color.Black);
        drawArea.DrawRectangle(pen, x, y, size, size);
    }

    // other methods and properties
}
```

Een klasse die de interface implementeert, kan op het scherm getoond worden

Voorbeeld

Het grote voordeel is dat het gedeelte van het programma dat vormen moet tekenen kan gebruik maken van polymorfie (en dus enkel op de hoogte moet zijn van de interface)

```
public class DrawingProgram
{
    private List<IDisplayable> shapes = new List<IDisplayable>();
    ...
    public void UpdateScreen()
    {
        foreach (IDisplayable shape in shapes)
        {
            shape.Display(myArea);
        }
    }
    ...
}
```

Demo

- Maak extra klasse Square die interface implementeert
- Maak lijst in Form1 van Ishape
- Voeg Square en Balloon object in lijst
- Maak button2 en laat alle elementen in lijst in 1 keer afdrukken

Toepassing van interfaces: `IList`

- Een `ListBox` control heeft achter de schermen een klasse die de items beheert
- Deze klasse is een implementatie van een interface `IList`

Toepassing van interfaces: IList

```
public interface IList
    : ICollection, IEnumerable
{
    int Add(object value);
    void Clear();
    bool Contains(object value);
    int IndexOf(object value);
    void Insert(int index, object value);
    void Remove(object value);
    void RemoveAt(int index);

    bool IsFixedSize { get; }
    bool IsReadOnly { get; }
    object Item(int index) { get; set; }
}
```

Toepassing van interfaces: IList

```
IList myList = shoppingListBox.Items;
```

Geeft als resultaat een instantie van ListBox.ObjectCollection terug

```
public class ObjectCollection :  IList
```

Verschil met abstracte klassen

1. Een abstracte klasse kan voor sommige methodes een implementatie voorzien, een interface kan dit nooit
2. Een klasse kan meerdere interfaces implementeren, maar slechts van één (abstracte) klasse overerven
   - Om over te erven hoeft de parent klasse niet abstract te zijn!
3. Een interface zorgt al tijdens compilatie voor controle, bij een abstracte klasse wordt er pas tijdens de uitvoering gekoppeld met een concrete methode/property
4. Een interface alleen wil nog niet zeggen dat deze zal gebruikt worden voor overerving

Dieren

Waarom?

- Dier -> Interface
- Vogel -> Abstracte klasse
- Papegaai -> Klasse

Oefening 1: ontwerpbeschrijving

- 23.2
  - Maak interface IBankrekening
  - Maak klasse Bankrekening die IBankrekening implementeert
  - Maak basic GUI
    - Gebruiker haalt bedrag uit Textbox
    - Knop voor storting of opname
    - Label met bedrag erin

Oefening 2: Dierentuin

- Zorg dat directeur van dierentuin een overzicht kan krijgen van elk dier en zijn bijhorende eigenschappen
- Aparte eigenschappen per categorie
  - Eet
  - Zegt

Oefening 3: Experimenteer met code CyberSpace Invader (Hfst 20 in boek)

- Voeg meerdere spelers toe
- Voeg verschillende soorten bommen toe
- …

Oefening 4: aansluiting

- 23.5
  - Maak interface IMyList (**Add**, *IndexOf, Remove*)
  - Maak klasse StringList die IMyList implementeert
    - IV: Array van string variabelen, int lengte van array
    - Overschrijving van methodes **Add**, IndexOf, Remove
    - Property P get om array terug te geven
  - Maak basic GUI
    - Textbox om tekst in te geven
    - Knop voor **Add**, Remove, IndexOf
    - Listbox om lijst in te zetten
  - Code achter GUI
    - IV van StringList a
    - Event achter knop
      - listBox1.DataSource = null; listBox1.DataSource = a.P

Kan je overerving nemen van
- Ilist
- List
- ArrayList