# Mechanizm dziedziczenia (rodzaje dziedziczenia)

Piotr Janowski

# Dziedziczenie w C#

Dziedziczenie w C# to mechanizm programowania obiektowego, który pozwala tworzyć nowe klasy na podstawie już istniejących. Dzięki temu można ponownie wykorzystać kod, rozwijać go i łatwo go rozszerzać.

## Podstawy

- **Klasa bazowa** (ang. *base class*) – udostępnia pola, właściwości i metody.
- **Klasa pochodna** (ang. *derived class*) – przejmuje elementy klasy bazowej i może dodawać własne.

# Jak działa dziedziczenie?

- Klasa **pochodna** (ang. *derived class*) przejmuje pola i metody z klasy **bazowej** (ang. *base class*).
- Można rozszerzać lub zmieniać zachowanie klasy bazowej poprzez dodawanie nowych elementów albo nadpisywanie metod.
- W C# stosujemy operator : do określenia, że klasa dziedziczy po innej:

*class Pies : Zwierze { }*

# Rodzaje dziedziczenia w C#

## Pojedyncze dziedziczenie

- Klasa pochodna może dziedziczyć tylko po jednej klasie bazowej.

Przykład:

```
class Zwierze { }
class Pies : Zwierze { }
```

## Dziedziczenie wielopoziomowe

- Klasa może dziedziczyć po innej klasie, która sama dziedziczy po jeszcze innej.

Przykład:

```
class Zwierze { }
class Ssak : Zwierze { }
class Czlowiek : Ssak { }
```

# Rodzaje dziedziczenia w C#

## Dziedziczenie hierarchiczne

- Kilka klas dziedziczy po tej samej klasie bazowej.

  Przykład:

  *class Zwierze { }*

  *class Pies : Zwierze { }*

  *class Kot : Zwierze { }*

## Dziedziczenie hybrydowe (kombinacja różnych rodzajów)

- Np. hierarchiczne + wielopoziomowe.
- W C# nie ma wielodziedziczenia klas (czyli klasa nie może dziedziczyć po więcej niż jednej klasie), ale można osiągnąć podobny efekt przez interfejsy.

# Przykłady dziedziczenia w C#

1. Dziedziczenie pojedyncze

```
using System;

class Zwierze
{
    public void Jedz()
    {
        Console.WriteLine("Zwierzę je.");
    }
}

class Pies : Zwierze
{
    public void Szczekaj()
    {
        Console.WriteLine("Pies szczeka: Hau hau!");
    }
}

class Program
{
    static void Main(string[] args)
    {
        Pies reksio = new Pies();
        reksio.Jedz();     // metoda z klasy bazowej
        reksio.Szczekaj(); // metoda z klasy pochodnej
    }
}
```

# Przykłady dziedziczenia w C#

2. Dziedziczenie wielopoziomowe

```
using System;

class Zwierze
{
    public void Oddychaj()
    {
        Console.WriteLine("Zwierzę oddycha.");
    }
}

class Ssak : Zwierze
{
    public void KarmMlekiem()
    {
        Console.WriteLine("Ssak karmi młode mlekiem.");
    }
}

class Czlowiek : Ssak
{
    public void Pracuj()
    {
        Console.WriteLine("Człowiek pracuje.");
    }
}

class Program
{
    static void Main(string[] args)
    {
        Czlowiek jan = new Czlowiek();
        jan.Oddychaj();   // z klasy Zwierze
        jan.KarmMlekiem();// z klasy Ssak
        jan.Pracuj();     // z klasy Czlowiek
    }
}
```

# Przykłady dziedziczenia w C#

3. Dziedziczenie hierarchiczne

```
using System;

class Zwierze
{
    public void Jedz()
    {
        Console.WriteLine("Zwierzę je.");
    }
}

class Pies : Zwierze
{
    public void Szczekaj()
    {
        Console.WriteLine("Pies szczeka.");
    }
}

class Kot : Zwierze
{
    public void Mialcz()
    {
        Console.WriteLine("Kot miauczy.");
    }
}

class Program
{
    static void Main(string[] args)
    {
        Pies reksio = new Pies();
        reksio.Jedz();
        reksio.Szczekaj();

        Kot filemon = new Kot();
        filemon.Jedz();
        filemon.Mialcz();
    }
}
```

# Przykłady dziedziczenia w C#

4. Dziedziczenie hybrydowe (interfejsy + klasy)

```
class Zwierze
{
    public void Oddychaj()
    {
        Console.WriteLine("Zwierzę oddycha.");
    }
}

interface ILatam
{
    void Lec();
}

interface IPlywam
{
    void Plywaj();
}

class Kaczka : Zwierze, ILatam, IPlywam
{
    public void Lec()
    {
        Console.WriteLine("Kaczka leci.");
    }

    public void Plywaj()
    {
        Console.WriteLine("Kaczka pływa.");
    }
}

class Program
{
    static void Main(string[] args)
    {
        Kaczka k = new Kaczka();
        k.Oddychaj(); // z klasy bazowej
        k.Lec();      // z interfejsu ILatam
        k.Plywaj();   // z interfejsu IPlywam
    }
}
```