Gajewski Andrzej

[Adres e-mail]

Streszczenie

[Przyciągnij uwagę czytelnika interesującym streszczeniem. Zazwyczaj jest to krótkie podsumowanie dokumentu.   
Aby dodać zawartość, wystarczy kliknąć tutaj i zacząć wpisywanie.]

Programowanie w C#

Programowanie obiektowe

Spis treści

[Klasy 2](#_Toc525045560)

[Klasa a obiekt 2](#_Toc525045561)

[Budowa klasy 2](#_Toc525045562)

[Elementy klasy 2](#_Toc525045563)

[Konstruktory 2](#_Toc525045564)

[Tablice i listy obiektów 2](#_Toc525045565)

[Właściwości 2](#_Toc525045566)

[Składniki statyczne 2](#_Toc525045567)

[Struktury a klasy 2](#_Toc525045568)

[Dziedziczenie 2](#_Toc525045569)

[Dziedziczenie – wprowadzenie 2](#_Toc525045570)

[Dziedziczenie po klasie System.Object 2](#_Toc525045571)

[Klasy pochodne 2](#_Toc525045572)

[Metody wirtualne 2](#_Toc525045573)

[Klasy abstrakcyjne 2](#_Toc525045574)

[Interfejsy 2](#_Toc525045575)

[Interfejsy - wprowadzenie 2](#_Toc525045576)

[Interfejsy a kolekcje (listy) 2](#_Toc525045577)

[Implementacja interfejsu 2](#_Toc525045578)

# Klasy

## Klasa a obiekt

## Budowa klasy

## Elementy klasy

## Konstruktory

## Tablice i listy obiektów

## Właściwości

## Składniki statyczne

## Struktury a klasy

# Dziedziczenie

## Dziedziczenie – wprowadzenie

## Dziedziczenie po klasie System.Object

## Klasy pochodne

## Metody wirtualne

Mówimy w programowaniu o dwóch rodzajach poliformizmu: statycznym i dynamicznym. W wersji dynamicznej wielopostaciowość udostępniana jest poprzez metody wirtualne - konkretna wersja metody może być ustalona dopiero w czasie wykonywania programu (na podstawie typu obiektu). Jeżeli z jakiegoś powodu chcemy udaremnić poliformizm, możemy zastosować operator *new*, który pozwala zerwać więzy z klasą bazową.

Metody wirtualne nie mogą być prywatne ani statyczne!!!

namespace MetodyVirtualne

{

public class Pojazd //Klasa bazowa

{

public virtual void Jedzie()

{

Console.WriteLine("Pojazd jedzie");

}

}

public class Samochod : Pojazd

{

public override void Jedzie()

{

Console.WriteLine("Samochód jedzie");

}

}

public class Pociag : Pojazd

{

public override void Jedzie()

{

Console.WriteLine("Pociąg jedzie");

}

}

class Program

{

static void Main(string[] args)

{

List<Pojazd> listaPojazdow = new List<Pojazd>();

listaPojazdow.Add(new Pojazd());

listaPojazdow.Add(new Samochod());

listaPojazdow.Add(new Pociag());

foreach(Pojazd p in listaPojazdow)

{

p.Jedzie();

}

Console.ReadKey();

}

}

}

## 

## 

## Klasy abstrakcyjne

# Interfejsy

## Interfejsy - wprowadzenie

## Interfejsy a kolekcje (listy)

## Implementacja interfejsu