Zespół Szkół Mechanicznych nr 1 im. Franciszka Siemiradzkiego
Technikum nr 10 Mechaniczne z Oddziałami Mistrzostwa Sportowego
Branżowa Szkoła I Stopnia nr 10 Mechaniczna
ul. Św. Trójcy 37, 85-224 Bydgoszcz
zsm1.bydgoszcz.pl

Materiały dydaktyczne

Przedmiot: Programowanie obiektowe/Pracownia programowania obiektowego
Opracował: Mirosław Miciak
Ćwiczenie: C6
Tematy: Podział programu na funkcje, Biblioteki języka programowania, Proste i złożone typy danych, Zmienne, Operacje na zmiennych: wejścia, wyjścia, arytmetyczne logiczne, Operacje na typach złożonych, Operatory arytmetyczne, przypisania, porównania, Operatory logiczne i bitowe, Operatory do obsługi łańcuchów, Instrukcja warunkowa i instrukcja wyboru, Instrukcje pętli, Wykorzystanie wybranej biblioteki matematycznej, Wykorzystanie bibliotek algorytmów, Funkcje, Klasa, obiekt, metoda, pole, Projektowanie prostych klas, Zmienne obiektowe, Zakres widoczności pól klasy, Metody klasy, Konstruktor i destruktor, Konstruktor kopiujący, Metody klasy - zakres widoczności, Prosta aplikacja obiektowa: konstruktor, destruktor, Składniki statyczne klasy, Klasy dziedziczone, Klasy zaprzyjaźnione, Dziedziczenie, Hermetyzacja, Polimorfizm, Funkcje zaprzyjaźnione z klasą, Metody do obsługi elementów statycznych klasy, Klasy bazowe i pochodne, Metody wirtualne, Klasy abstrakcyjne, Szablony klas dla prostych typów liczbowych, Obsługa wyjątków, Instrukcja throw, Obsługa błędów wykonania aplikacji.

Konstruktor

Konstruktor to specjalna metoda, która jest wywoływana podczas tworzenia obiektu danej klasy. Konstruktor ma tę samą nazwę co klasa i może być przeciążany. Głównym zadaniem konstruktora jest inicjalizacja obiektu.

```
public class Osoba
{
    public string Imie { get; set; }
    public int Wiek { get; set; }

    // Konstruktor domyślny
    public Osoba()
    {
        Imie = "Nieznane";
        Wiek = 0;
    }

    // Konstruktor przeciążony
    public Osoba(string imie, int wiek)
    {
        Imie = imie;
        Wiek = wiek;
    }

    public void Wyswietl()
    {
        Console.WriteLine($"Imię: {Imie}, Wiek: {Wiek}");
    }
```

```
}

public class Program
{
    public static void Main(string[] args)
    {
        Osoba osoba1 = new Osoba();
        osoba1.Wyswietl();

        Osoba osoba2 = new Osoba("Jan", 30);
        osoba2.Wyswietl();
    }
}
```

Destruktor

Destruktor to metoda specjalna, która jest wywoływana, gdy obiekt jest usuwany z pamięci. W C# destruktory są rzadko używane, ponieważ .NET Framework zarządza pamięcią za pomocą mechanizmu garbage collection. Destruktory są używane do uwalniania zasobów niezarządzanych.

```
public class Zasob
{
    // Konstruktor
    public Zasob()
    {
        Console.WriteLine("Zasób utworzony.");
    }

    // Destruktor
    ~Zasob()
    {
        Console.WriteLine("Zasób zniszczony.");
    }

    public void UzyjZasob()
    {
        Console.WriteLine("Używanie zasobu...");
    }
}

public class Program
{
    public static void Main(string[] args)
    {
        Zasob zasob = new Zasob();
        zasob.UzyjZasob();
    }
}
```

Konstruktor kopiujący

Konstruktor kopiujący to specjalny konstruktor, który jest używany do kopiowania zawartości jednego obiektu do innego. Jest przydatny, gdy chcemy utworzyć nowy obiekt jako kopię istniejącego.

```
public class Punkt
{
    public int X { get; set; }
```

```
    public int Y { get; set; }

    // Konstruktor domyślny
    public Punkt(int x, int y)
    {
        X = x;
        Y = y;
    }

    // Konstruktor kopiujący
    public Punkt(Punkt punkt)
    {
        X = punkt.X;
        Y = punkt.Y;
    }

    public void Wyswietl()
    {
        Console.WriteLine($"X: {X}, Y: {Y}");
    }
}

public class Program
{
    public static void Main(string[] args)
    {
        Punkt punkt1 = new Punkt(10, 20);
        Punkt punkt2 = new Punkt(punkt1); // Użycie konstruktora kopiującego

        punkt1.Wyswietl();
        punkt2.Wyswietl();
    }
}
```

Pokażmy bardziej złożony przykład, który obejmuje konstruktor, destruktor i konstruktor kopiujący, a także ilustruje bardziej zaawansowane zarządzanie zasobami. Załóżmy, że mamy klasę KontoBankowe, która zarządza danymi konta, takimi jak numer konta, saldo oraz historię transakcji. Zademonstrujemy konstruktor, destruktor i konstruktor kopiujący w kontekście tej klasy.csharp

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

// Klasa reprezentująca Konto Bankowe
public class KontoBankowe
{
    public int NumerKonta { get; set; }
    public double Saldo { get; set; }
    public List<string> HistoriaTransakcji { get; set; }

    // Konstruktor domyślny
    public KontoBankowe()
    {
        NumerKonta = 0;
        Saldo = 0.0;
        HistoriaTransakcji = new List<string>();
        Console.WriteLine("Utworzono Konto Bankowe.");
```

```
    }

    // Konstruktor przeciążony
    public KontoBankowe(int numerKonta, double saldo)
    {
        NumerKonta = numerKonta;
        Saldo = saldo;
        HistoriaTransakcji = new List<string>();
        Console.WriteLine($"Utworzono Konto Bankowe nr {numerKonta} z początkowym saldem {saldo}.");
    }

    // Konstruktor kopiujący
    public KontoBankowe(KontoBankowe inneKonto)
    {
        NumerKonta = inneKonto.NumerKonta;
        Saldo = inneKonto.Saldo;
        HistoriaTransakcji = new List<string>(inneKonto.HistoriaTransakcji);
        Console.WriteLine($"Skopiowano Konto Bankowe nr {inneKonto.NumerKonta}.");
    }

    // Metoda do dodawania transakcji
    public void DodajTransakcje(string opis, double kwota)
    {
        Saldo += kwota;
        HistoriaTransakcji.Add($"{opis}: {kwota}");
    }

    // Metoda do wyświetlania historii transakcji
    public void WyswietlHistorie()
    {
        Console.WriteLine($"Historia transakcji dla konta nr {NumerKonta}:");
        foreach (var transakcja in HistoriaTransakcji)
        {
            Console.WriteLine(transakcja);
        }
    }

    // Destruktor
    ~KontoBankowe()
    {
        Console.WriteLine($"Zniszczono Konto Bankowe nr {NumerKonta}.");
    }
}

public class Program
{
    public static void Main(string[] args)
    {
        // Utworzenie konta za pomocą konstruktora przeciążonego
        KontoBankowe konto1 = new KontoBankowe(12345, 1000.0);
        konto1.DodajTransakcje("Wpłata", 500.0);
        konto1.DodajTransakcje("Wypłata", -200.0);
        konto1.WyswietlHistorie();

        // Utworzenie kopii konta za pomocą konstruktora kopiującego
        KontoBankowe konto2 = new KontoBankowe(konto1);
        konto2.DodajTransakcje("Wpłata", 300.0);
```

```
        konto2.WyswietlHistorie();

        // Domyślny destruktor będzie wywołany automatycznie przy zakończeniu programu
    }
}
```

Opis
KontoBankowe
Posiada właściwości NumerKonta, Saldo i HistoriaTransakcji.
Konstruktor domyślny inicjalizuje puste konto.
Konstruktor przeciążony inicjalizuje konto z podanym numerem i saldem.
Konstruktor kopiujący kopiuje dane z innego konta, w tym historię transakcji.
Destruktor wypisuje wiadomość przy usuwaniu obiektu.
Metody DodajTransakcje i WyswietlHistorie zarządzają transakcjami i wyświetlają historię.
Program
Tworzy konto bankowe, dodaje transakcje i wyświetla historię.
Kopiuje konto za pomocą konstruktora kopiującego, dodaje kolejną transakcję i wyświetla historię.

Zadania do samodzielnego rozwiązania

1. Zadanie: System Zarządzania Biblioteką
   Utwórz klasę Ksiazka z właściwościami Tytul, Autor oraz RokWydania.
   Zaimplementuj trzy różne konstruktory:
   Konstruktor domyślny, który ustawia Tytul na "Nieznany", Autor na "Nieznany", a RokWydania na 0.
   Konstruktor przeciążony, który przyjmuje jako parametry Tytul, Autor i RokWydania, a następnie ustawia te właściwości.
   Konstruktor kopiujący, który kopiuje dane z innego obiektu Ksiazka.
   Dodaj metodę WyswietlInformacje, która wyświetla informacje o książce.
   Dodaj destruktor, który wyświetli komunikat "Zniszczono obiekt Ksiazka: {Tytul}".
   Utwórz klasę Biblioteka, która przechowuje listę obiektów Ksiazka.
   Dodaj metody DodajKsiazke oraz UsunKsiazke, które dodają i usuwają książki z listy.
   Dodaj metodę WyswietlKsiazki, która wyświetla wszystkie książki w bibliotece.
   Utwórz klasę Program i zaimplementuj metodę Main, która:
   Tworzy kilka obiektów Ksiazka za pomocą różnych konstruktorów.
   Dodaje te książki do obiektu Biblioteka.
   Wyświetla wszystkie książki w bibliotece.
   Usuwa jedną książkę z biblioteki.
   Ponownie wyświetla wszystkie książki w bibliotece.

2. Utwórz klasę Book z właściwościami Title i Author. Napisz trzy różne konstruktory: domyślny, przecią żony i kopiujący. Przetestuj je w metodzie Main.

3. Dodaj destruktor do klasy Book, który wyświetli komunikat "Book object destroyed." Sprawdź działan ie destruktora.

4. Utwórz klasę Rectangle z właściwościami Length i Width. Zaimplementuj konstruktor przeciążony ora z konstruktor kopiujący. Napisz metodę CalculateArea, która obliczy pole prostokąta i przetestuj ją.