Konstruktory

Przegląd zagadnień

- Konstruktory: o co chodzi ?
- Wywołanie innego konstruktora
- Inicjalizatory obiektów
- Kolejność inicjalizacji
- Struktury: inicjalizacja i konstruktory
- Destruktor
- Wzorce projektowe
- Tworzenie obiektu na podstawie prototypu

Konstruktor: wstęp

- Metoda klasy
- Metoda wywoływana tuż po utworzeniu obiektu
- Posiada taką samą nazwę jak klasa w której jest zdefiniowany
- Nie określamy wartości zwracanej
- Wywoływany przez operator new
- Można go przeciążyć

Konstruktor: przykład

```
class NazwaKlasy{
   public NazwaKlasy(){ //konstruktor bezargumentowy
      ...
   }
   public NazwaKlasy(int arg1,double arg2){
                        //konstruktor z argumentami
      ...
   }
}
```

```
NazwaKlasy x1 = new NazwaKlasy();
NazwaKlasy x1 = new NazwaKlasy(2, 3.3);
```

Konstruktor bezargumentowy

- Generowany automatycznie przez kompilator, pod warunkiem, że nie ma innego konstruktora
- Inne nazwy
  - domyślny
  - domniemany

Wywołanie innego konstruktora

- Lista inicjalizacyjna konstruktora
- Nie wolno wywoływać rekurencyjnie

```
class NazwaKlasy{
  private Typ1 pole1;
  private Typ2 pole2;
  public NazwaKlasy(): this(war1){    }
  public NazwaKlasy(Typ1 arg1):this(arg1, war2){    }
  public NazwaKlasy(Typ1 arg1, Typ2 arg2) {
    pole1 = arg1;
    pole2 = arg2;
  }
}
```

Inicjalizatory obiektów: przykład

```
class K1{
      public int A;
      public int B;
}
class K2{
      public int X;
      public int Y;
      public K1 K;
}
```

```
K2 x = new K2(){
                  X=20,
                  K = new K1(){
                              A =20,
                              B=30
                             }
               };
K1 k1 = new K1(){A =20, B=30};
```

Inicjalizatory obiektów: opis

- Nie wszystkie pola muszą być inicjalizowane w inicjalizatorze
- Wcześniej jest wywoływany konstruktor domyślny, chyba że wskażemy inny
- Inicjalizatory mogą być zagnieżdżone
- Można inicjalizować klasy i struktury

Kolejność inicjalizacji

- Pola są inicjalizowane wartością zero dla danego typu
- Nadawane są wartości przypisane w miejscu definicji
  - `private int nazwaPola = 10;`
- Wartość przypisana w konstruktorze.
  - Wartości przypisane w konstruktorze wywołanym na liście inicjalizacyjnej są nadpisywane
- Wartości przypisane przy pomocy inicjalizatora obiektów

Struktury: różnice

- Typ wartości
- Nie można deklarować konstruktora domyślnego
- Wszystkie pola muszą być zainicjalizowane w konstruktorze
- Konstruktor domyślny jest zawsze generowany
- Nie można nadawać polu wartości w miejscu definicji
- Konstruktor domyślny inicjalizuje wszystkie pola wartością zero odpowiedniego typu
- Pola nie są domyślnie inicjalizowane wartością zero odpowiedniego typu

Destruktor

- W programowaniu obiektowym metoda wywoływana tuż przed zwolnieniem pamięci przez obiekt
- W języku C# - Finalizator
  - ~NazwaKlasy() { }
  - Metoda Finalize
  - Interfejs Idisposable
  - jest niedeterministyczny, wywoływany przez GarbageColector przed zwolnieniem pamięci zajmowanej przez obiekt
  - staramy się go unikać!!!

Wzorce projektowe

- Zbiór sprawdzonych w praktyce uniwersalnych rozwiązań, często pojawiających się rzeczywistych problemów projektowych.
- Podstawowy podział wzorców
  - Wzorce konstrukcyjne (ang. creational design patterns) – skupiają się procesie na tworzenia obiektu.
  - Wzorce strukturalne (ang. structural design patterns) – skupiają się budowie oraz organizacji klas i obiektów. Pomagają łączyć obiekty w większe struktury.
  - Wzorce czynnościowe (ang. behavioral design patterns) – skupiają się na opisie interakcji obiektów i klas oraz podziale odpowiedzialności między klasami

Tworzenie obiektu na podstawie prototypu

- Rodzaje kopi
  - kopia płytka
  - kopia głęboka
- Konstruktora, który przyjmuje jako parametr zmienną typu określonego przez klasę, w której jest zdefiniowany
- Metoda MemberwiseClone
- Interfejs IClonable
- Wzorzec projektowy prototyp