

# Projektowanie obiektowe oprogramowania

## Zestaw A

### Inversion of Control (2)

2018-05-15

Liczba punktów do zdobycia: **6/68**

Zestaw ważny do: 2018-05-29

*Uwaga! Kontynuacja pracy nad silnikiem Inversion of Control na identycznych zasadach. W szczególności obowiązkową częścią każdego zadania są testy jednostkowe, nawet jeśli nie wspomina się o tym w treści zadań.*

#### 1. (1p) (Wstrzykiwanie instancji)

Silnik IoC rozbudować o możliwość rozwiązywania zadanej instancji danego typu. Uzupełnia to katalog możliwości predefiniowania o zwracanie konkretnych instancji (już zaimplementowane metody rozwiązywania zostają).

Uwaga, w przypadku sekwencyjnej rejestracji typu przez `RegisterType/RegisterInstance` kontener zachowuje się jak do tej pory, czyli obowiązuje zasada *ostatnia rejestracja jest wiążąca*.

```
public class SimpleContainer
{
    public void RegisterType<T>( bool Singleton ) where T : class;
    public void RegisterType<From, To>( bool Singleton ) where To : From;

    // nowe!
    public void RegisterInstance<T>( T Instance );
}

SimpleContainer c = new SimpleContainer();

IFoo foo1 = new Foo();
c.RegisterInstance<IFoo>( foo );

IFoo foo2 = c.Resolve<IFoo>();

// foo1 == foo2
```

#### 2. (5p) (Dependency Injection)

Silnik IoC rozbudować o mechanizm Dependency Injection, zakładając że wstrzykiwaniu podlegają na razie tylko konstruktory.

To oznacza, że metoda `Resolve` kontenera nie musi już zakładać, że obiekt ma konstruktor bezparametrowy.

Zamiast tego kontener ogląda sygnatury konstruktorów, wybiera konstruktor o możliwie najdłuższej liczbie parametrów (lub konstruktor oznaczony atrybutem `[DependencyConstructor]`, pod warunkiem że jest tylko jeden taki) i próbuje rekursywnie rozwikływać obiekty będące

parametrami konstruktora, budując w ten sposób drzewo rozwikłań. Taki proces kontynuuje się schodząc w dół drzewa, konsekwentnie rozwikłując konstruktory kolejnych obiektów, za każdym rozwikłaniem korzystając z wiedzy jaką ma kontener (czyli dostarczonej informacji o zarejestrowanych typach / instancjach).

*Uwaga. W przypadku dwóch konstruktorów o tej samej, maksymalnej liczbie parametrów można zachować się na trzy sposoby:*

- wyrzucić wyjątek,
- próbować rozwikływać którykolwiek konstruktor,
- próbować rozwikływać wszystkie konstruktory o maksymalnej liczbie parametrów po kolei, aż do błędu lub udanego rozwikłania któregoś z nich.

*Uwaga. Podczas rozwikływania może dojść do sytuacji powstania cyklu w drzewie (najprostszyp przypadk: obiekt A w konstruktorze żąda obiektu typu A. Kontener powinien wykryć taką sytuację i zareportować ją zrozumiałym wyjątkiem.*

Przykład 1:

```
public class A
{
    public B b;

    public A( B b )
    {
        this.b = b;
    }
}

public class B { }

SimpleContainer c = new SimpleContainer();
A a = c.Resolve<A>();

// kontener wykonstruuje a typu A z wstrzykniętą instancją B. Test [a.b != null] przechodzi.
```

Przykład 2:

```
public class X
{
    public X( Y d, string s ) { };
}

public class Y { }

SimpleContainer c = new SimpleContainer();
X x = c.Resolve<X>();

// wyjątek, string nie ma konstruktora bezparametrowego i nie da się rozwikłać żadnego z konstruktorów

// ... ale ....

c.RegisterInstance( "ala ma kota" ); // rejestruje instancję string
X x = c.Resolve<X>();

// jest ok, zarejestrowano instancję string więc rozwikłanie konstruktora X jest możliwe
```

Wiktor Zychła