

# Projektowanie obiektowe oprogramowania

## Zestaw 4

Wzorce podstawowe i kreacyjne

2014-03-18

Liczba punktów do zdobycia: **5/30**

Zestaw ważny do: 2014-04-15

*Uwaga! Obowiązkową częścią każdego rozwiązania są testy jednostkowe.*

1. **(1p) (Singleton)** Przygotować implementacje singletonów o następujących politykach czasu życia:

- jedna instancja dla całego procesu
- jedna osobna instancja dla każdego wątku
- jedna instancja na co najwyżej 5 kolejnych sekund

Dostarczyć właściwe testy jednostkowe.

2. **(1p) (Factory)** Zaimplementować fabrykę obiektów **dowolnego** typu, parametryzowaną nazwą typu i flagą określającą czy obiekt ma być singletonem. Dostarczyć właściwe testy jednostkowe.

```
public class GenericFactory
{
    public object CreateObject(
        string TypeName,
        bool IsSingleton,
        params object[] Parameters );
}

// klient
GenericFactory factory = new GenericFactory();
var a1 = factory.CreateObject( "System.Collections.ArrayList", false );
var o1 = factory.CreateObject( "UWr.Examples.ExampleType", true );
```

3. **(1p) (Open Factory)** Powtórzyć przykład z wykładu fabryki otwartej na rozszerzenia:

```
public class ShapeFactory
{
    public RegisterWorker( IShapeFactoryWorker worker ) {
        ...
    }

    public IShape CreateShape( string ShapeName, params object[] parameters ) {
        ...
    }
}

// klient
```

```
ShapeFactory factory = new ShapeFactory();
IShape square = factory.CreateShape( "Square", 5 );

// rozszerzenie
factory.RegisterWorker( new RectangleFactoryWorker() );
IShape rect = factory.CreateShape( "Rectangle", 3, 5 );
```

4. **(1p) (Object Pool)** Zaimplementować klasę **Airport** dostarczającą instancje obiektów typu **Plane**. Założyć że lotnisko dysponuje ograniczoną liczbą samolotów. Poprawnie obsłużyć przypadki realizowane przez pulę: pobieranie, zwalnianie, przekroczenie rozmiaru puli.

Dostarczyć właściwe testy.

5. **(1p) (Builder)** Przykład z wykładu

<http://netpl.blogspot.com/2012/02/simple-fluent-and-recursive-tag-builder.html>

rozwinąć o obsługę wcięć (z wcięciami/bez wcięć/głębokość wcięć), po to żeby można było napisać:

```
StringWriter writer = new StringWriter(...);
TagBuilder tag      = new TagBuilder( writer );
tag.IsIndented      = true;
tag.Indentation     = 4;
tag.StartTag( "parent" )
    .AddAttribute( "parentproperty1", "true" )
    .AddAttribute( "parentproperty2", "5" )
    .StartTag( "child1" )
        .AddAttribute( "childproperty1", "c" )
        .AddContent( "childbody" )
        .EndTag()
    .StartTag( "child2" )
        .AddAttribute( "childproperty2", "c" )
        .AddContent( "childbody" )
        .EndTag()
    .EndTag()
    .StartTag( "script" )
        .AddContent( "$scriptbody();" )
        .EndTag();
```

Dostarczyć właściwe testy.

Wiktor Zychla