Projektowanie obiektowe oprogramowania
Zestaw 5

Wzorce strukturalne

2016-04-05

Liczba punktów do zdobycia: **6/36**
Zestaw ważny do: 2016-04-19

1. (**1p**) (**Facade**) Dostarczyć implementacji dla fasady:

```
class SmtpFacade {
public void Send( string From, string To,
                        string Subject, string Body,
                        Stream Attachment, string AttachmentMimeType );
}
```

   W implementacji użyć klas z biblioteki standardowej, realizujących wysyłkę za pomocą SMTP.

2. (**1p**) (**Decorator**) Zaimplementować klasę strumienia `CaesarStream`, który będzie działał jak dekorator strumieni (wymaganym parametrem konstruktora będzie inny strumień, na którym operował będzie `CaesarStream`) przy czym odczytując i zapisując dane będzie aplikował przesunięcie cezariańskie poszczególnych bajtów danych. Przykład użycia:

```
FileStream fileToWrite = File.Create( ... );
CaesarSteram caeToWrite = new CaesarStream( fileToWrite, 5 );
// 5 to przesunięcie

caeToWrite.Write( ... );

FileStream fileToRead = File.Open( ... );
CaesarStream caeToRead = new CaesarStream( fileToRead, -5 );
// -5 znosi 5

caeToRead.Read( ... );
```

3. (**1p**) (**Proxy**) Dostarczyć implementacji proxy ochraniającego dla którejś klasy z poprzedniego zestawu, którego mechanizm ochrony polega na tym, że interfejs obiektu jest dostępny tylko w godzinach 8-22, w pozostałych godzinach korzystanie z obiektu powoduje wyrzucanie wyjątków.

   Dostarczyć drugiego proxy logującego, które loguje wywołania metod (data, nazwa metody i wartości parametrów) oraz zakończenia wywołań (data, wartość zwracana z metody).

4. (**1p**) (**Adapter**) Dostarczyć adapter rozwiązujący zadanie:

   `http://netpl.blogspot.com/2009/05/c-puzzle-no15-intermediate.html`

5. (**2p**) (**Bridge**) Dana jest klasa rejestru pracowników.

```
public class PersonRegistry
{
    /// <summary>
    /// Pierwszy stopień swobody - różne wczytywanie
    /// </summary>
    public List<Person> _persons = new List<Person>();

    /// <summary>
    /// Drugi stopień swobody - różne użycie
    /// </summary>
    public void NotifyPersons()
    {
        foreach ( Person person in _persons )
            Console.WriteLine( person );
    }
}

public class Person { }
```

Wiadomo też, że że w wyniku analizy funkcjonalnej zidentyfikowano dwa stopnie swobody tej klasy.

Jeden polega na możliwości wczytywania rejestru z różnych źródeł (XML, baza danych). Drugi polega na możliwości powiadamiania pracowników w różny sposób (mail, SMS).

Przygotować **dwie** różne implementacje mostu (każda za jeden punkt), w których jeden ze zidentyfikowanych stopni swobody stanie się wstrzykiwaną implementacją, a drugi pozostanie jako możliwy do pokrycia w podklasie klasy abstrakcji (rejestru).

Wiktor Zychla