Programowanie C#

Zadanie 8

Piotr Błaszyński

5 maja 2019

Przygotować aplikację w C# pozwalającą na przetestowanie prostej implementacji drzewa binarnego. W MS Visual Studio.NET 2017 (jak również w wersjach wcześniejszych od 2008) jest dostępna możliwość napisania i uruchomienia testów jednostkowych dla swojego programu. Żeby przygotować testy jednostkowe (ang. unit tests) należy po uruchomieniu VS.NET wybrać Create->New Project->Visual C#->Test->Unit Test Project (zwrócić uwagę, żeby lokalizacja była zaufana, bo inaczej może być problem z uruchamianiem testów). W przykładzie korzystamy z domyślnie dostepnego narzedzia od Microsoftu, ale w praktyce polecam xUnit (możliwe, że będzie konieczna instalacja np. przy pomocy narzędzia NuGet). W pliku UnitTest1.cs znajdują się przykładowe testy jednostkowe, dla naszych potrzeb najważniejsze są dwa atrybuty:

```
[TestClass]
[TestMethod]
```

oraz metoda z klasy Assert:

```
Assert.AreEqual(expected, actual);
```

inna, bogatsza wersja pozwalającą na podanie delty i wyświetlanego komunikatu:

```
public static void AreEqual(double expected, double
   actual, double delta, string message);
```

Pierwszy z atrybutów mówi o tym, że to jest klasa testująca, przy pomocy kolejnego oznacza się metodę testową. Przy wykorzystaniu tych dwóch atrybutów, klasy Assert i testów z wcześniejszych zajęć należy przygotować zestaw testów. Następnie należy przygotowany zestaw odpalić (prawy przycisk myszy->Run test).

Czasami wykorzystywanym (praktyka pokazuje, że częściej polecanym niż wykorzystywanym) podejściem do tworzenia oprogramowania jest TDD (Test Driven

Development), którego istotnym elementem jest podejście Red->Green->Refactor. Polega ona na tym, że najpierw piszemy testy (na początku jeden, góra dwa), testy się nawet nie kompilują poprawnie, więc mamy Fail testów (faza Red), następnie uzyskujemy wynik jak najprostszym sposobem (na początku nawet lekko oszukując), dopisujemy kolejne testy, które na początku co prawda już się powinny kompilować, ale wyniki testowanej klasy powinny (mogą, w przypadku prostych klas nie muszą) być inne od oczekiwanych (prawidłowych wyników). Po doprowadzeniu do sytuacji, że wszystkie testy się wykonują poprawnie (faza Green), przeprowadza się refaktoryzacje (refaktoring, ang. refactoring) czyli ulepszenie kodu bez zmiany jego działania. Dzięki działającym testom, możliwe jest wykonywanie zmian w kodzie bez obaw, że zepsuje się działającą wcześniej funkcjonalność.

Dla zainteresowanych tematem kilka linków do poczytania:
https://martinfowler.com/books/refactoring.html
http://davesquared.net/2009/01/introduction-to-solid-principles-of-oo.html
http://en.wikipedia.org/wiki/Occam%27s_razor
https://en.wikipedia.org/wiki/SOLID
http://butunclebob.com/ArticleS.UncleBob.PrinciplesOfOod