28 marca 2019

Programowanie obiektowe

Lista 4.

Za każde zadanie można otrzymać do 4 pkt, jednak można oddać nie więcej niż 2 zadania. Zadania wykonaj w C#. Tradycyjnie do każdego zadania powinien być dołączony krótki program ilustrujący wykorzystanie zaimplementowanych klas i metod.
Uwaga Powyższe zadania należy wykonać nie wykorzystując klas bibliotecznych. Można jednak korzystać z tablic.

Zadanie 1. Wybierz zaprogramowane wcześniej przez Ciebie dwie kolekcje. Zastanów się, jakie są wspólne operacje w tych kolekcjach. Zaprogramuj interfejs zawierający nagłówki tych operacji i przebuduj tak implementacje tych kolekcji, aby klasy implementowały ten interfejs. Dodatkowo zaimplementuj w jednej z tych kolekcji interfejs `IEnumerable`, metodę `string ToString()`, dostęp indeksowany i właściwość `int Length`.

Zadanie 2. Zaprogramuj klasę ***PrimeCollection*** implementującą interfejs `IEnumerable` (bądź `IEnumerable<T>`) omówiony na wykładzie. Obiekty tej klasy powinne być kolekcją liczb pierwszych. Jednak kolekcja ta nie powinna budować prawdziwej kolekcji, tylko "w locie" obliczać kolejną liczbę pierwszą. Przykładowo poniższy program

```
PrimeCollection pc = new PrimeCollection();
foreach(int p in pc)
      Console.Writeline(p);
```

powinien wypisać kolejne liczby pierwsze aż do momentu przekroczenia zakresu typu `int`.

Zadanie 3. Zaimplementuj dwie klasy implementujące różne sposoby reprezentacji grafu nieskierowanego; może to być np. reprezentacja macierzowa oraz reprezentacja za pomocą list sąsiedztwa (ale muszą być różne). Przyjmij, że węzły są etykietowane wartościami typu `string`. W każdej klasie zdefiniuj metodę generowania losowego grafu o zadanej liczbie węzłów i krawędzi. Zadeklaruj interfejs zawierający podstawowe metody i własności potrzebne do obsługiwania grafu. Interfejs ten powinien być implementowany przez obydwie klasy.

Następnie zaprogramuj dowolny algorytm wyszukiwania najkrótszej drogi między dwoma węzłami grafu wskazanymi za pomocą etykiet. Zadbaj o to, aby w algorytmie odwoływać się tylko do metod zadeklarowanych w interfejsie. Zmierz czasy wykonania tego algorytmu dla różnych reprezentacji grafu.

Zamiast przeszukiwania możesz też zaimplementować w obydwu klasach odpowiednie metody, dzięki którym grafy staną się prawdziwymi kolekcjami wierzchołków lub krawędzi, które można przetwarzać instrukcją `foreach`. Wybór, czy graf jest kolekcją wierzchołków czy krawędzi należy uzasadnić przy oddawaniu programu.

Zadanie 4. Zaprojektuj i zaimplementuj odpowiedni zbiór klas do reprezentowania produkcji gramatyk bezkontekstowych. Zaimplementuj metodę generowania losowych słów wyprowadzanych w tej gramatyce.

Marcin Młotkowski