16 marca 2023

Programowanie obiektowe

Lista 4.

Za każde zadanie można otrzymać do 4 pkt, jednak można oddać nie więcej niż 2 zadania. Zadania wykonaj w $C^\sharp$. Tradycyjnie do każdego zadania powinien być dołączony krótki program ilustrujący wykorzystanie zaimplementowanych klas i metod.

Zadanie 1
Wybierz zaprogramowane wcześniej przez Ciebie dwie kolekcje. Zastanów się, jakie są wspólne operacje w tych kolekcjach. Zaprogramuj

- interfejs ***ListCollection*** zawierający nagłówki tych operacji i przebuduj tak implementacje tych kolekcji, aby klasy implementowały ten interfejs;
- w jednej z tych kolekcji zaprogramuj metody interfeju `IEnumerable`, metodę `string ToString()`, dostęp indeksowany i właściwość `int Length`.

Zadanie 2
Zaprogramuj klasę ***SlowaFibonacciego***, która generuje *słowa Fibonacciego* (obiekty klasy string) z dwóch liter: a i b. Słowa te definiujemy następująco:

$$S_n = \begin{cases} b & \text{gdy } n = 1 \\ a & \text{gdy } n = 2 \\ S_{n-1} \bullet S_{n-2} & \text{gdy } n > 2 \end{cases}$$

gdzie $S_{n-1} \bullet S_{n-2}$ oznacza konkatenację słów S_{n-1} i S_{n-2}. Klasa powinna implementować omówiony na wykładzie interfejs `IEnumerable` (bądź `IEnumerable<T>`) tak, aby przykładowy program

```
SlowaFibonacciego sf = new SlowaFibonacciego(6);
foreach(string s in sf)
    Console.Writeline(s);
```

wypisał

```
b
a
ab
aba
abaab
abaababa
```

Argument konstruktora jest liczbą słów, jakie ma zwrócić obiekt. Można przyjąć, że obiekt nie przechowuje całej listy wcześniej zwróconych słów.

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 ***IGraph*** zawierający podstawowe metody i własności potrzebne do obsługiwania grafu. Interfejs ten powinien być implementowany przez obydwie klasy.
Zaprogramuj klasę ***GraphOperations*** z dwiema metodami:

- `void createRandom(IGraphg g, int vert, int edges)` która tworzy losowy graf z `vert` wierzchołkami i liczba krawędzi `edges`. Obiekt `g` powinien być pustym grafem, tj. pustym zbiorem krawędzi i wierzchołków.

- `List<Vertex> shortestPath(IGraph g, Vertex a, Vertex b)` która zwraca najkrótszą ścieżkę z wierzchołka o etykiecie `a` to `b`.

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