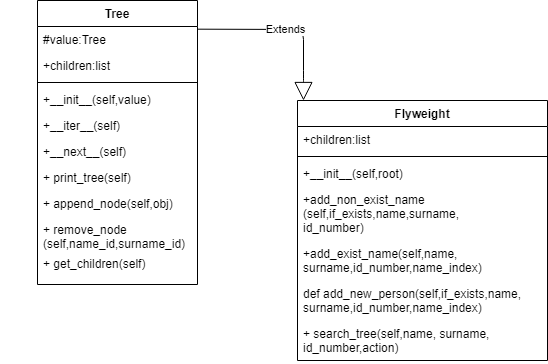
|  |  |  |
| --- | --- | --- |
| Aleksandra Duda | Gr. 31 lab 02 | Techniki Obiektowe |
| Nr indeksu: 130510 | Laboratorium nr 4 | 10.11.20r. |

**Sprawozdanie z labolatorium nr 4**

1. **Cel ćwiczenia**

Wykonanie programu przedstawiającego ewidencję ludności.

1. **Model UML**



1. **Przebieg ćwiczenia**

* W programie mieliśmy użyć struktury drzewa, a więc najpierw zaimplementowałam klasę **Tree.**  Ta klasa była odpowiedzialna za tworzenie drzewa, które będzie miało poziomy takie jak: Imię, Nazwisko i Pesel.

Tworzymy obiekt przy pomocy konstruktora nadając mu początkową wartość **self.value**. W konstruktorze jest również zaincjalizowana pusta tablica **self.children.**

Klasa **Tree** zawiera między innymi metodę **append\_node,** która służy dodawaniu węzła do drzewa, jest również operacja usuwania z drzewa **remove\_node**.

W **Tree** został zaimplementowany niestandardowy **iterator** dzięki któremu możemy budować drzewo z utworzonych danych.

Funkcja **print\_tree** korzysta z zaimplementowanych metod **\_\_iter\_\_** i **\_\_next\_\_.**

Funkcja **get\_children** zwraca ilość potomków.



* Aby uniknąć redundancji danych należało zaimplementować wzorzec Flyweight. **Flyweight** jest klasą mającą metody, które sprawdzają, czy jakieś dane już się pojawiły – jeśli nie to tworzy je, a jeśli tak to resztę danych dopisuje już do istniejących. Klasa **Flyweight** pilnuje także, żeby do drzewa nie można było dodać osoby o istniejącym już tym drzewie peselu.

Aby żadne dane nie mogły się powtarzać musiałam zaimplementować algorytm przeszukiwania drzewa **search\_tree**.

Ta metoda przyjmuje argumenty takie jak name,surname,id\_number i action.

Metoda przeszukuje elementy w drzewie i sprawdza, czy jakiś element podany w argumencie funkcji istnieje w drzewie. Jeśli tak to zmienną imienia, naziska lub peselu ustawiamy na **True,** jeśli nie ma podanej wartości w drzewie to ustawiamy na **False.**

W przypadku gdy chcemy dodać nową osobę do drzewa to ta tablica prawdy jest przekazywana do funkcji **add\_new\_person**. W tej metodzie następnie jest sprawdzane, czy dane imie się pojawiło i jeśli tak to nazwisko i pesel są dopisywane juz do istniejącego imienia(przekazanie wykonania zadania do funkcji **add\_exist\_name**). Natomiast jeśli numer pesel istnieje już w drzewie to niemożliwym jest „utworzenie nowej osoby”. Jeśli natomiast dane imię nie istniało wcześniej to przy pomocy funkcji **add\_non\_exist\_name** możemy je stworzyć.

Możemy również tworzyć osoby z nazwiskami dwuczłonowymi, które są zapisywane w postaci listy.

Jednak gdy chcemy usunąć daną osobę to wykorzystujemy algorytm przeszukiwania – w momencie,gdy osoba zostanie znaleziona (pokrywa się imię, nazwisko i pesel w drzewie z szukanymi danymi) to z algorytmu przeszukiwania są przekazywane indeksy imienia, nazwiska i nr pesel do funkcji **remove\_node.** Ta funkcja jest w klasie **Tree** , ale z racji tego, że **Flyweight**  dziedziczy po **Tree**  to ma do tej metody dostęp.

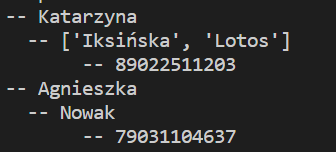




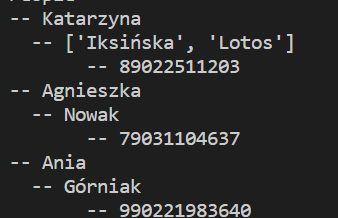
1. **Wnioski**

* **Iterator** znacznie usprawnił tworzenie drzewa i wydaje się być to nawet prostszym rozwiązaniem. Dzięki temu implementujemy mniej kodu i użycie iteratora wpływa na zoptymalizowanie algorytmu.
* Dzięki klasie **Flyweight** mogliśmy zapobiec powtarzaniu się danych oraz tworzeniu osoby, która już istnieje, czyli takiej, która ma dokładnie taki sam nr pesel.
* Klasa **Flyweight** sprawia, że Imiona i pesele nie powtarzają się w całym drzewie, a nazwiska w danym węźle.
* Zrobiłam kilka testów aby sprawdzić, czy program przeze mnie napisany działa jak należy:

Początkowo przygotowałam dane przykładowe:

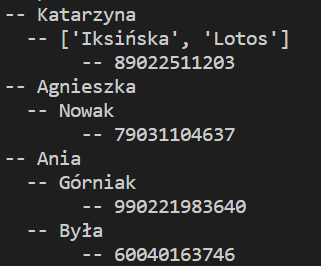


Natępnie dodałam nową osobę, która nie istnieje w drzewie – Annę Górniak o nr pesel 990221983640.



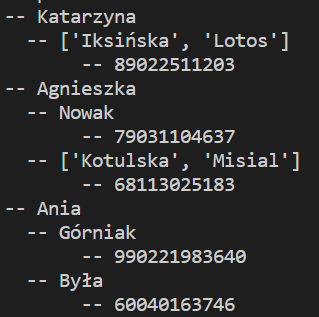
Jak widać osoba została dodana „niezależnie”, bo imię nie istniało wcześniej w drzewie.

Następnie dodaję osobę o imieniu, które już jest w drzewie. Tym razem niech to będzie „Ania Była”

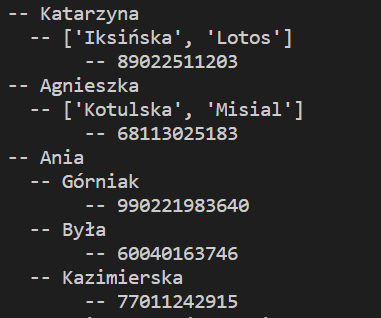


Jak widzimy, nazwisko i nr pesel dodały się już do istniejącego imienia.

Podobnie stało się w przypadku dodania do bazy „Agnieszki Kotulskiej – Misial”.



Następnie postanowiłam usunąć z bazy „Agnieszkę Nowak”.



Jak widać cały węzeł się usunął.

Struktura drzewa w każdym z tych przypadków budowała się jak należy i zgodnie z oczekiwaniami.

Szczególnie pomocną okazała mi się operacja sprawdzania na początku, czy dany pesel już istnieje, bo to znacznie przyspiesza proces - gdybyśmu tego nie sprawdzali to w drzewie pojawiałyby się osoby z dokładnie takimi samymi danymi, a gdybyśmy sprawdzali nr pesel, ale dopiero na końcu to musielibyśmy usuwać już powstałe dane.

Małe problemy występowały w nazwiskach dwuczłonowych, więc postanowiłam, że w tym przypadku posłużę się listą.