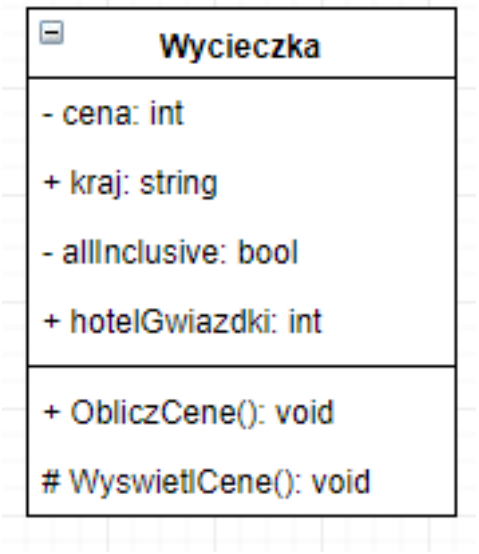
Ćwiczenia 2 - Powtórzenia z programowania obiektowego 2

B1. Stwórz w programie dwie klasy `Pies` i `WlascicielPsa`. Pies musi być wyprowadzony na spacer jeśli spożywał posiłek więcej niż 5 godzin wcześniej. W klasie Pies dodaj metody: * `UstawCzasPosilku(int)`, która ustawia ile godzin temu pies zjadł posiłek; * `PobierzCzasPosilku()`, która pobiera czas posiłku; * `PotrzebnySpacer()`, która zwraca true gdy pies musi być wyprowadzony na spacer.

W klasie `WlasicielPsa` dodaj metodę `ZabierzNaSpacer(Pies)`, która zwraca true gdy pies (będący parametrem metody) musi być wyprowadzony na spacer. W metodzie `Main` przetestuj działanie powyższych metod.

B2. Na podstawie diagramu UML stwórz klasę. Następnie w klasie `Program` stwórz co najmniej trzy obiekty, na których wykonasz co najmniej po jednym razie każdą metodą i przypiszesz obiektowi co najmniej jedno pole (możesz stworzyć brakujące metody, ale nie możesz modyfikować modyfikatorów dostępu na diagramie).



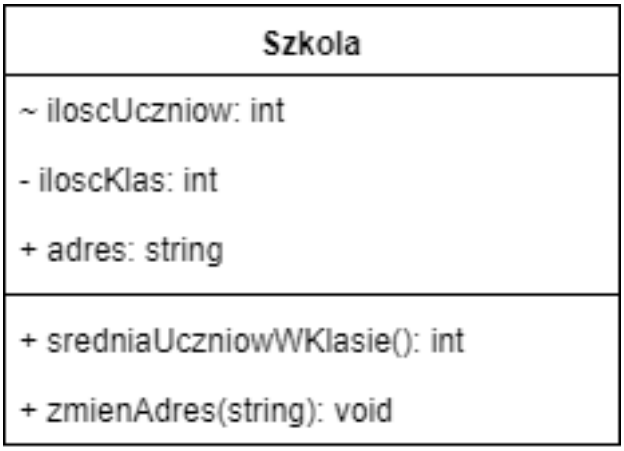
Rysunek 1: Polecenie2

B3. Na podstawie diagramu UML stwórz klasę. Następnie w klasie `Program` stwórz co najmniej trzy obiekty, na których wykonasz co najmniej po jednym razie każdą metodą i przypiszesz obiektowi co najmniej jedno pole (możesz stworzyć brakujące metody, ale nie możesz modyfikować modyfikatorów dostępu na diagramie).

B4. Stwórz klasę `Liczba`. Dodaj w niej publiczne pole `wartosc` typu `int` i przypisz mu wartość początkową `0`. Stwórz publiczne metody typu `void` o nazwach `Dodaj` i `Odejmij`, każda z nich ma być z jednym parametrem typu `int`, ich zadaniem jest odpowiednio powiększenie lub pomniejszenie pola `wartosc` o podanych parametr. Następnie w klasie `Program` stwórz kilka obiektów i przetestuj działanie metod.

B5. Puste (był błąd).

B6. Stwórz klasę na przechowywania informacji o dacie (nie korzystaj z systemowych "gotowców"). Zastanów się ile pól jest potrzebnych, jakie metody. Narysuj diagram UML klasy. Stwórz kilka obiektów. Następnie w klasie związanej z data stwórz metodę bez parametru, która na podstawie przechowywanej daty zwróci `string` z dniem tygodnia (np. wtorek, tu jedynie możesz skorzystać z systemowych metod).



Rysunek 2: Polecenie3

B7. Stwórz klasę `Kwaterniony`, dodaj w niej odpowiednia pola i metody tak aby mieć możliwość wykonywania arytmetyki na kwaternionach.

B8. Napisz program, który zawiera klasę `Opowiadanie` reprezentującą dowolny ciąg znaków o długości < 500. Klasa ta ma mieć prywatną tablicę składową `wyrazy`, prywatną składową `ile` i konstruktory: * bezargumentowy : wpisujący do tablicy `wyrazy` 345 znaków 'x', `ile = 100`, * dwuargumentowy : pierwszy argument to wzór znaku, a drugi to ilość powtórzeń. Potem do tablicy `wyrazy` należy dodać zgodnie z parametrami określoną ilość znaków. * jednoargumentowy : argumentem jest tablica znaków zawierająca łańcuch znaków (krótszy niż 500 znaków) – tekst ten jest kopiowany do tablicy `wyrazy`, ustalana jest wartość `ile`.

Ponadto klasa `Opowiadanie` ma zawierać metodę `ZnajdzZamien`, która ma 2 parametry typu `char`. We wnętrzu dodaj instrukcje wykonujące zamianę wszystkie wystąpień pierwszego parametru na drugi. Dalej bezargumentową metodę `Wyswietl` wyświetlającą tekst z tablicy `wyrazy`.

W klasie `Program` przetestuj działanie metod i tworzenie obiektów.