Vehicles-2

Wersja: 1.0

W tym zadaniu rozszerzysz funkcjonalność programu *Vehicles-1* o mechanizm generowania unikalnych wewnętrznych ID pojazdów.

1. VIN

Ćwiczenie na: składowe statyczne

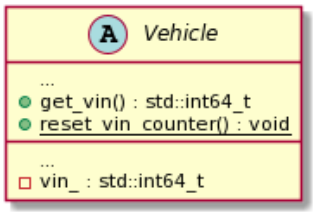
1.1 Polecenie

Przyjmij, że każdy unikalny obiekt klasy potomnej dla `Vehicle` (czyli utworzony *nie* przy użyciu konstruktora kopiującego) powinien posiadać unikalny numer VIN nadawany w chwili konstruowania takiego obiektu. Pierwszy pojazd powinien otrzymać numer VIN równy `1`.

Dodaj do klasy `Vehicle` pole `vin_` typu `std::int64_t`[1] oraz metody:

- `get_vin()` – zwraca wartość pola `vin_`
- `reset_vin_counter()` – resetuje licznik numerów VIN (najbliższa utworzona instancja klasy `Vehicle` znów otrzyma numer 1, kolejna – 2 itd.)

Nowa funkcjonalność została przedstawiona na rys. 1.



Rysunek 1. Składowe klasy `Vehicle` służące obsłudze numeru VIN

1.2 Wskazówki

Utwórz w klasie `Vehicle` pomocnicze pole statyczne `next_vin_`, którego wartość będzie inkrementowana (tj. zwiększana o 1) w konstruktorze – wykorzystaj to pole do nadawania numeru VIN (pole `vin_`) konkretnym obiektom.

Metoda `reset_vin_counter()` jest niezbędna do wykonania testów jednostkowych, gdyż testy są wykonywane w losowej kolejności i nie mielibyśmy pojęcia, jaka powinna być oczekiwana wartość zwrócona przez wywołanie metody `Vehicle::get_vin()` dla danego obiektu.

2. Domyślne implementacje metod specjalnych

Ćwiczenie na: `default`

Zdefiniuj jawnie, za pomocą słowa kluczowego **`default`**, destruktor klasy `Vehicle` oraz konstruktory kopiujące wszystkich trzech klas (`Vehicle`, `Car` i `Bicycle`).

[1] Więcej odnośnie typów całkowitych o stałej szerokości: C – Type - What are uint8_t, uint16_t, uint32_t and uint64_t? (badprog.com)