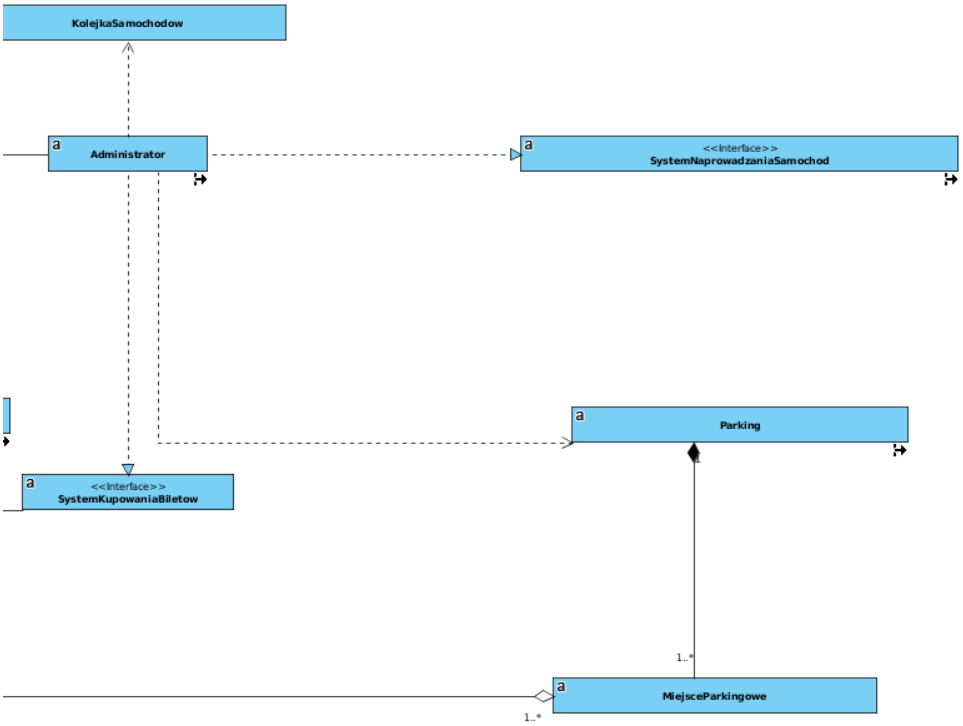
MODEL ZARZĄDZANIA PARKINGIEM SAMOCHODOWYM

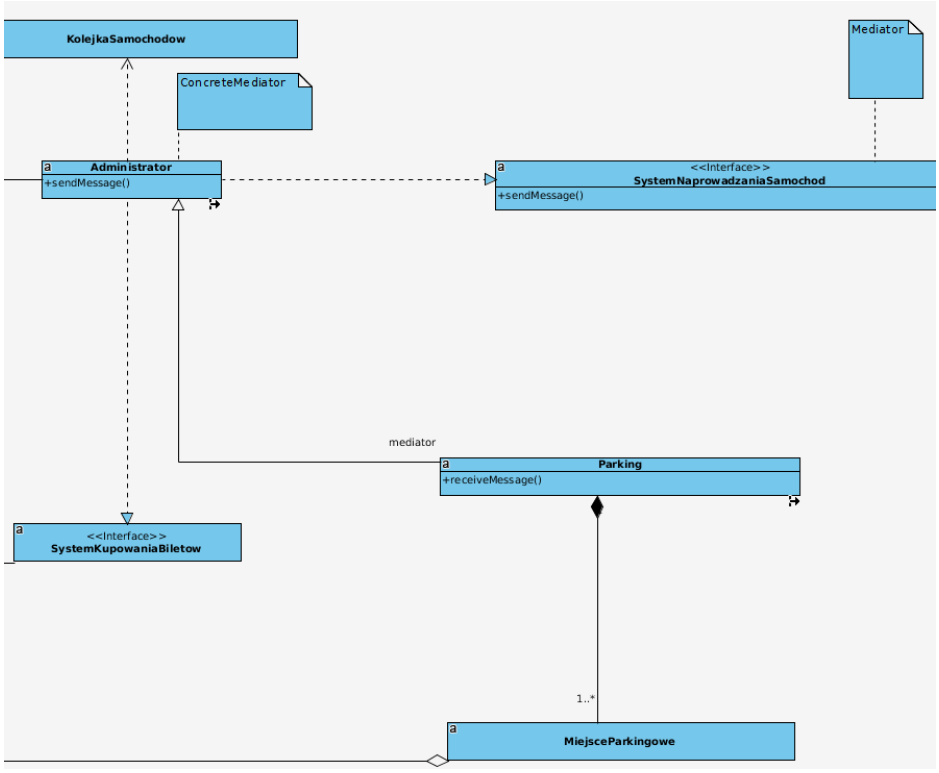
Czaja Hubert, Białek Tomasz, gr.1

Zarządzanie parkingiem samochodowym wymaga scentralizowanego mechanizmu komunikacji aby poprawnie obsługiwać klientów. Komunikacja ta na różnych obszarach parkingu musi być zawsze być poprawna i niezależna od reszty systemu, dlatego w naszym projekcie postanowiliśmy wykorzystać wzorzec **Mediator**.

Fragment diagramu klas przed wykorzystaniem **Mediatora**:



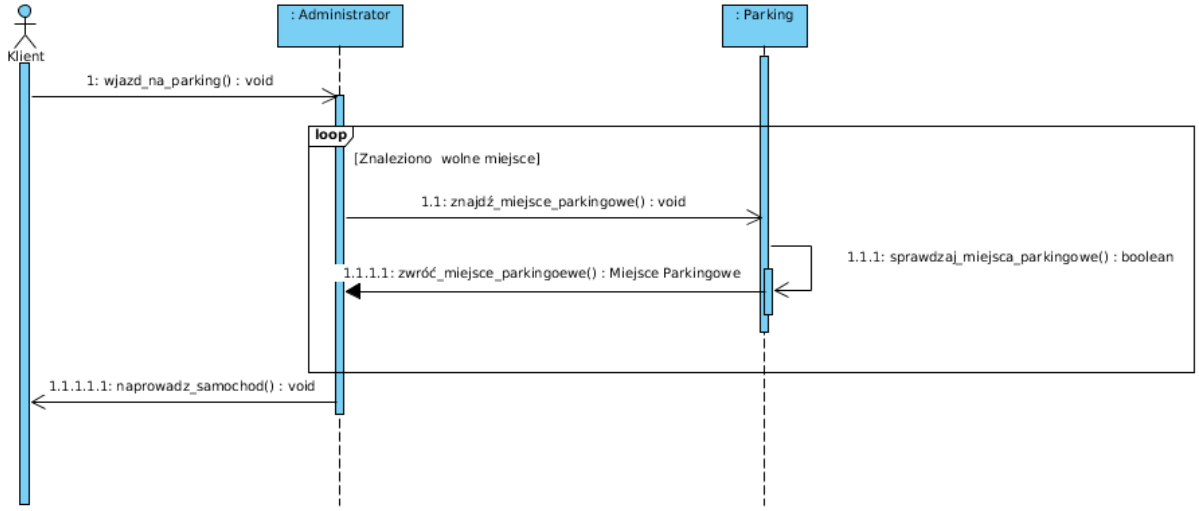
Fragment diagramu klas po wykorzystaniu **Mediatora**:



Chcemy aby *„SystemNaprowadzaniaSamochodów”* definiował interfejs do komunikacji z obiektami klasy „*Parking*”. „*Administrator*” implementuje więc takie zachowanie kooperatywne. Dzięki metodzie „*sendMessage()*” będzie mógł się komunikować z obiektami typu „*MiejsceParkingowe*” poprzez „*Parking*” co znacznie uprości komunikację między tymi komponentami.

Mediator odnajduje zastosowanie w naszym projekcie w sytuacji gdy wiele obiektów typu „*MiejsceParkingowe*” musi komunikować się ze sobą w celu wykonania określonego zadania np. znalezienia takiego miejsca, aby system naprowadzenia samochodów mógł naprowadzić na nie klienta.

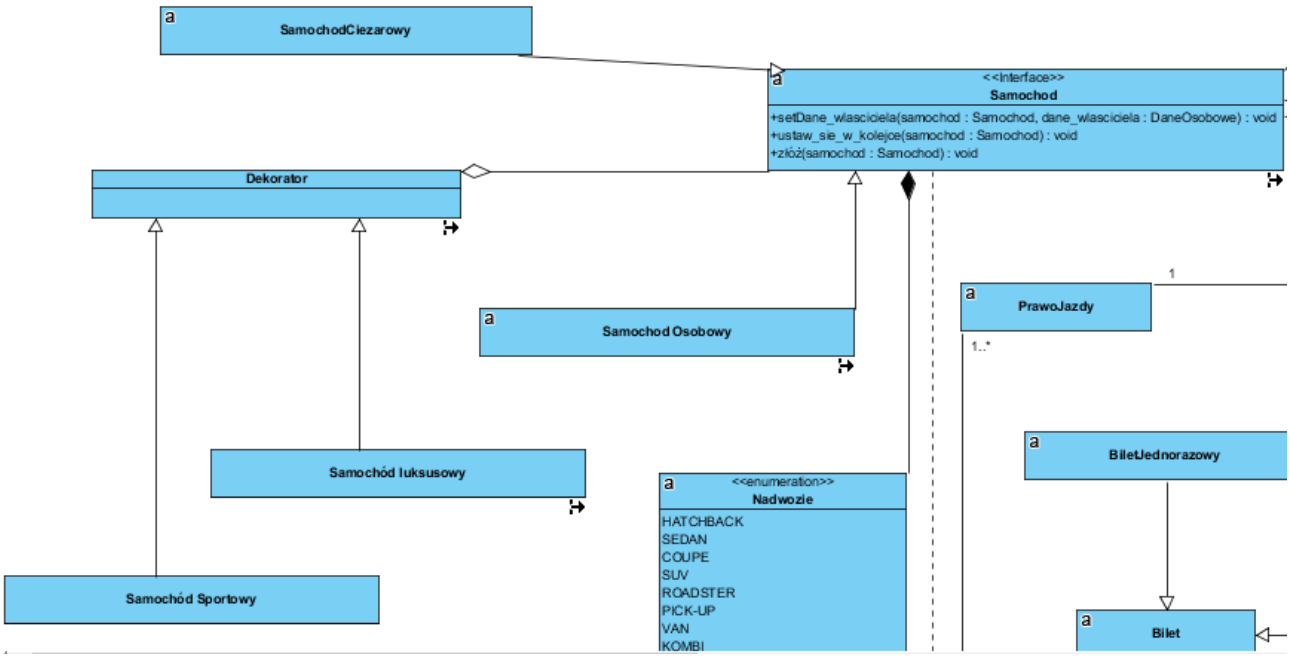
Fragment diagramu sekwencji przy wykorzystaniu **Mediatora**:



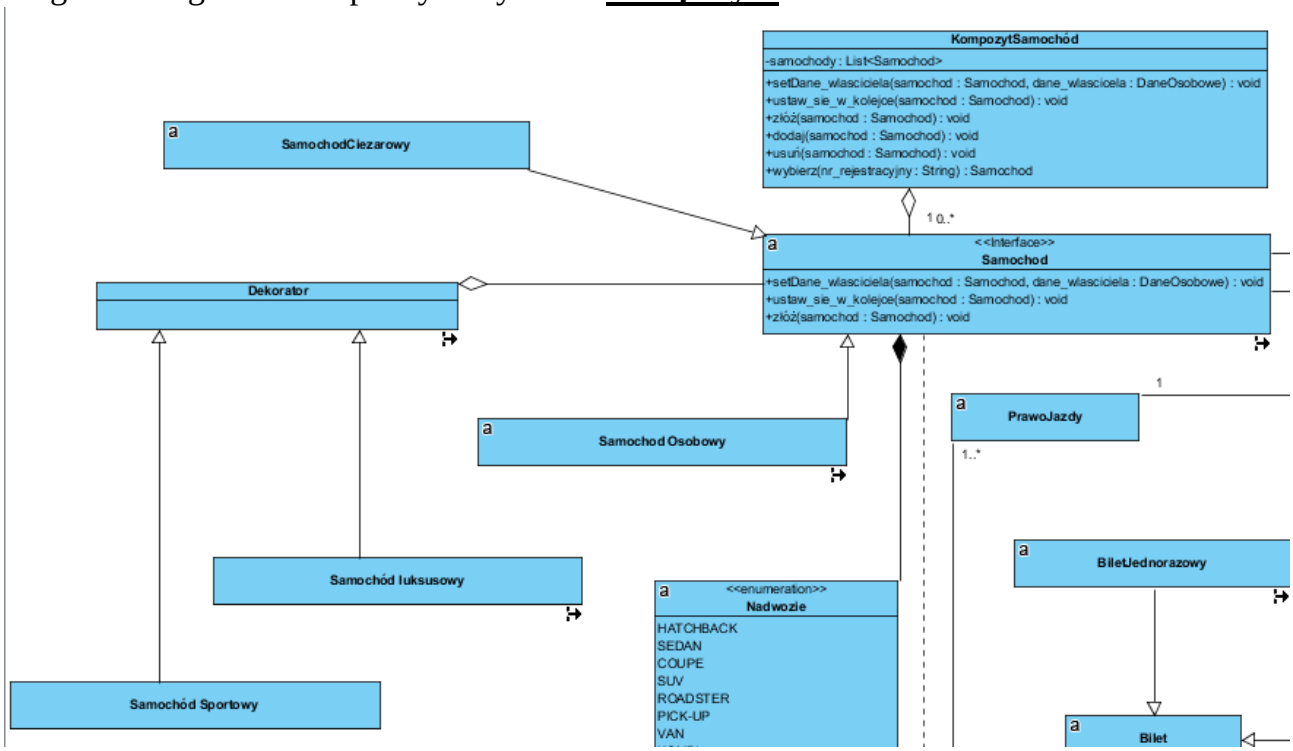
Klient w pierwszej kolejności po pojawieniu się na parkingu wysyła sygnał do administratora, aby ten znalazł mu wolne miejsce. „*Administrator*” jako **Mediator** komunikuje się z obiektami typu „*Miejsce Parkingowe*” poprzez „*Parking*” aby znaleźć odpowiednie miejsce. Wprowadzenie tego wzorca do naszego projektu jest o tyle istotne, że nie chcemy aby doszło do sytuacji, gdzie klient został wprowadzony w błąd i został pokierowany na nieodpowiednie miejsce. Mogłoby się to wydarzyć w pierwszej wersji systemu, gdzie nie było zdefiniowanego scentralizowanego punktu komunikacji i wyniku jakiegoś błędu w systemie mogłoby dojść do nieoczekiwanych problemów z obsługą klienta.

Kolejnym zastosowanym wzorcem w przykładzie parkingu samochodowego jest wzorzec projektowy o nazwie Kompozyt. Wzorzec ten umożliwia traktowanie pojedynczych obiektów jako coś większego – zbiór, którym można zarządzać.

Fragment diagramu klas przed wykorzystaniem **Kompozytu**:

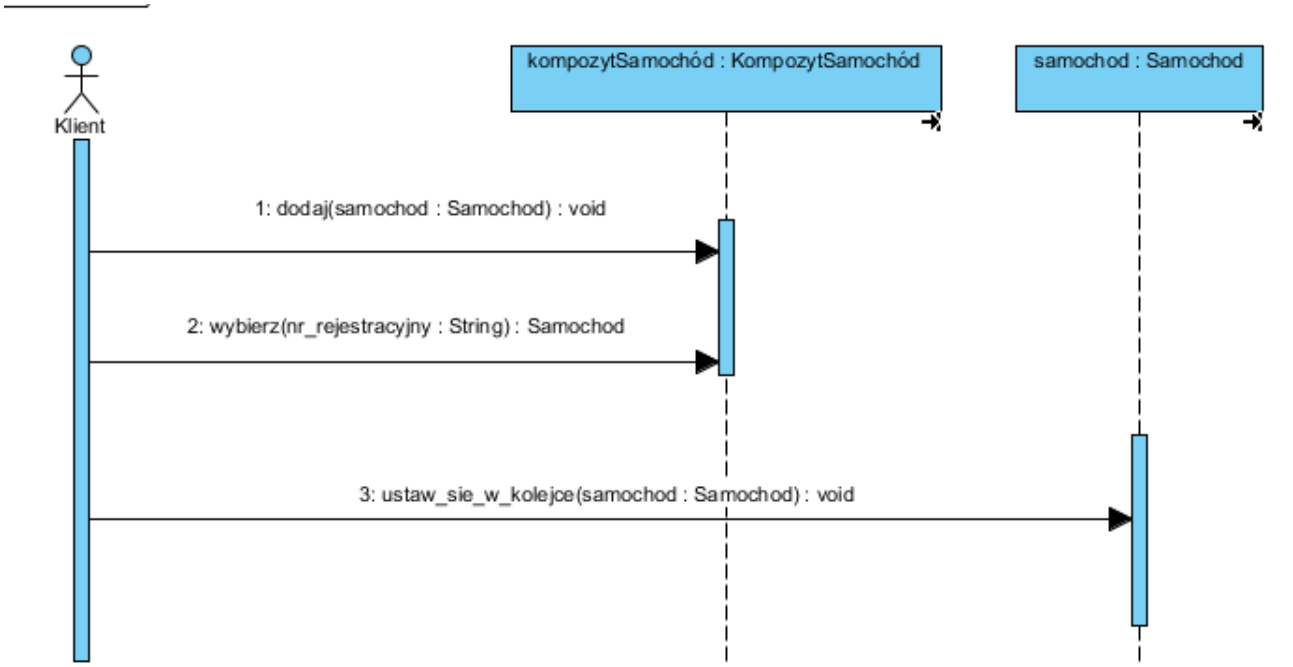


Fragment diagramu klas po wykorzystaniu **Kompozytu**:



Naszym komponentem jest interfejs „Samochód”. Klasa „KompozytSamochód” implementujeten intefejs oraz zawiera np. listę z typem „Samochód”. W klasie „KompozytSamochód” znajdują się także metody, które jako argument przyjmują typ „Samochód”, co pozwala na szeroką gamę możliwości – np. metoda „dodaj” umożliwia dodanie obiektu dowolnej klasy, która implementuje interfejs „Samochód”. W ten sposób otrzymujemy listę, która przechowuje np. obiekty klasy „SamochódOsobowy” czy też „SamochódLuksusowy”.

Przykładowy schemat diagramu sekwencji przy wykorzystaniu **Kompozytu**:



Jest to przykładowy przebieg wydarzeń, ponieważ przy wykorzystaniu wzorca kompozytu klient może robić dowolne rzeczy (te, które są zdefiniowane w interfejsie „Samochód") z obiektem. W tym przypadku klient po pojawieniu się na parkingu wywołuje metodę „dodaj" na klasie „KompozytSamochod", która dodaje samochód na listę. Następnie ze względu na unikalny numer rejestracyjny może on pobrać obiekt samochodu z listy, a następnie ustawić go w kolejce do obsłużenia.