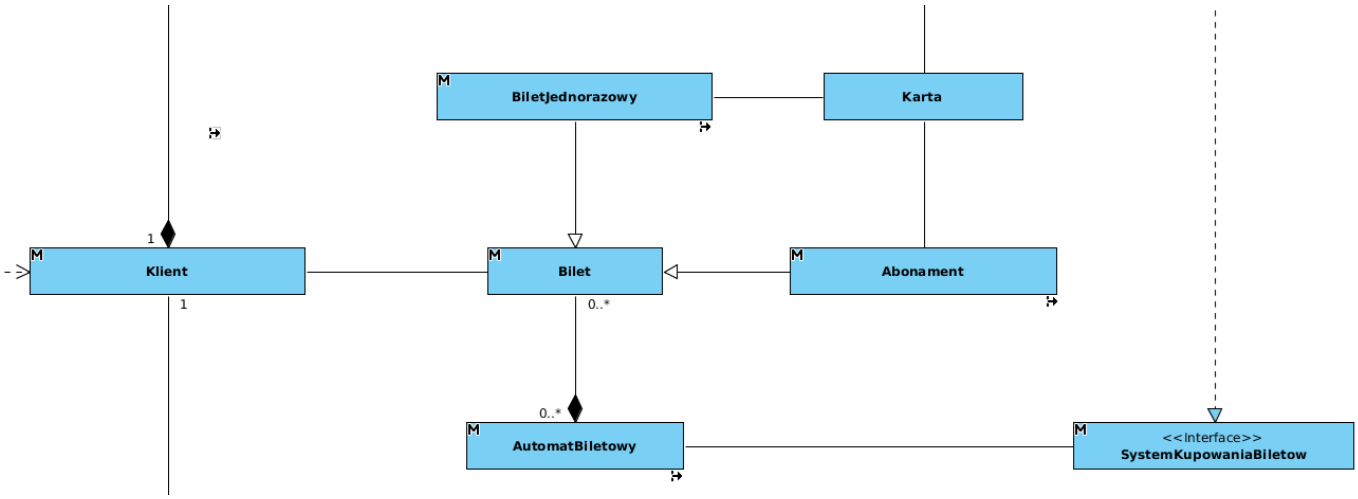
MODEL ZARZĄDZANIA PARKINGIEM SAMOCHODOWYM

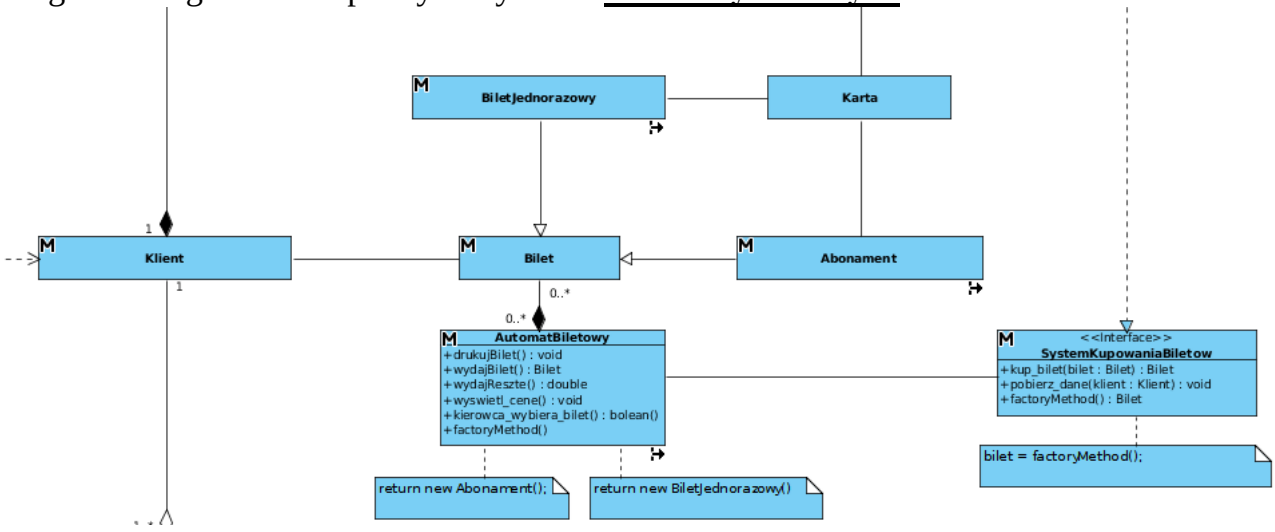
Czaja Hubert, Białek Tomasz, gr.1

W naszym projekcie kluczowymi elementami są systemy do naprowadzania samochodów oraz do kupowania biletów. Z logicznego punktu widzenia, automat biletowy, który umożliwia zakup abonamentu lub biletu jednorazowego powinien udostępniać jedynie interfejs w postaci jakiegoś terminala dla klienta parkingu. Tworzenie fizycznych obiektów powinno zostać ukryte. Odpowiednim rozwiązaniem tego problemu wydaje się wykorzystanie kreacyjnego wzorca projektowego **metody wytwórczej**. Klasa „*SystemKupowaniaBiletów*" powinna korzystać ze swojej podklasy „*AutomatBiletowy*" do specyfikacji obiektów, które tworzy. Wykorzystanie **metod wytwórczych** umożliwia klientom korzystanie z klasy „*AutomatBiletowy*" do tworzenia poszczególnych obiektów klas typu „*Bilet*". Tworzenie obiektów wewnątrz klas **metod wytwórczych** jest bardziej elastyczne niż tworzenie obiektów wprost. Kierowca wjeżdżający na parking nie powinien przecież widzieć całego procesu drukowania biletu.

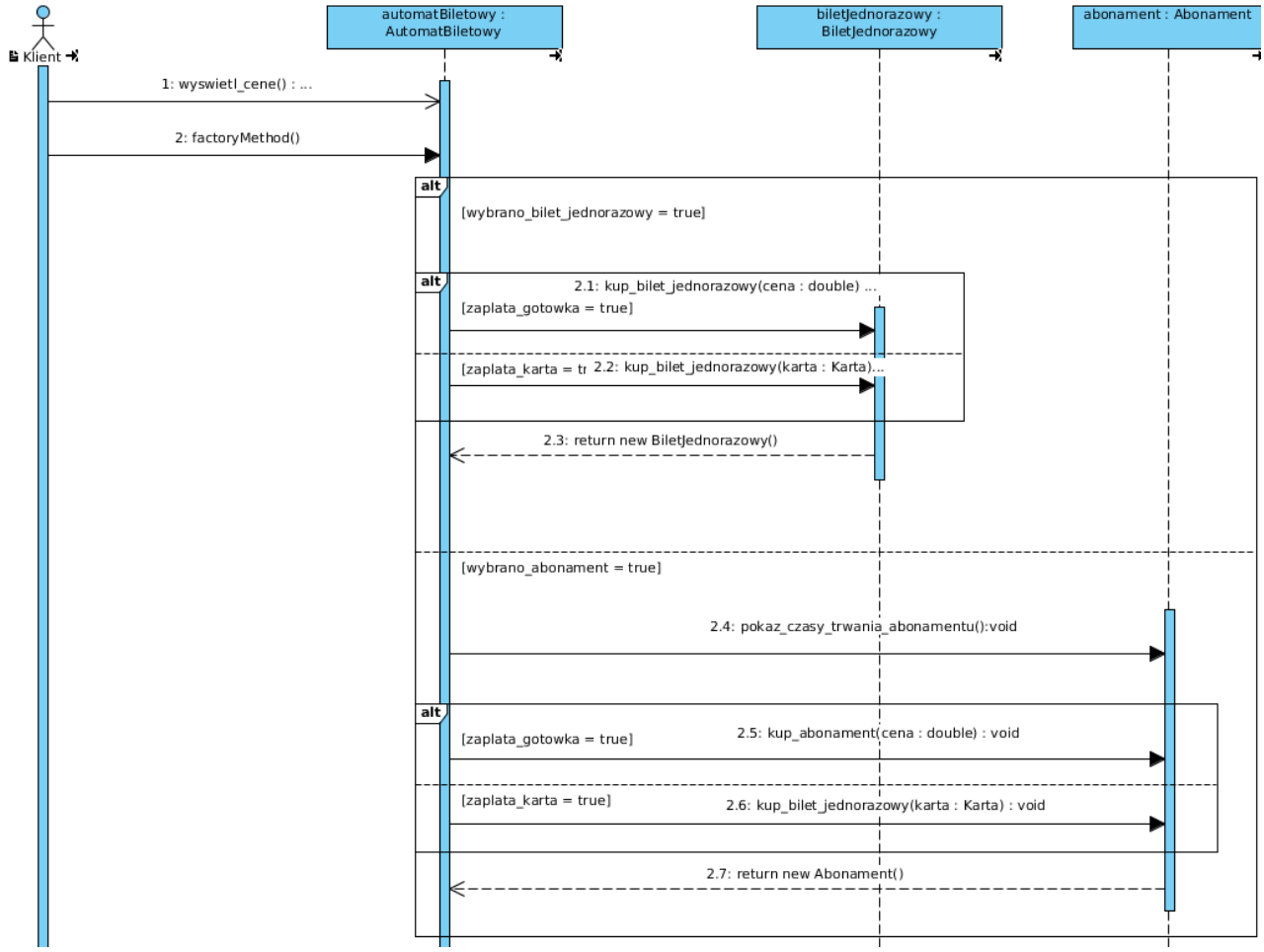
Fragment diagramu klas przed wykorzystaniem **metod wytwórczych**:



Fragment diagramu klas po wykorzystaniu **metod wytwórczych**:



Przebieg interakcji obiektów:

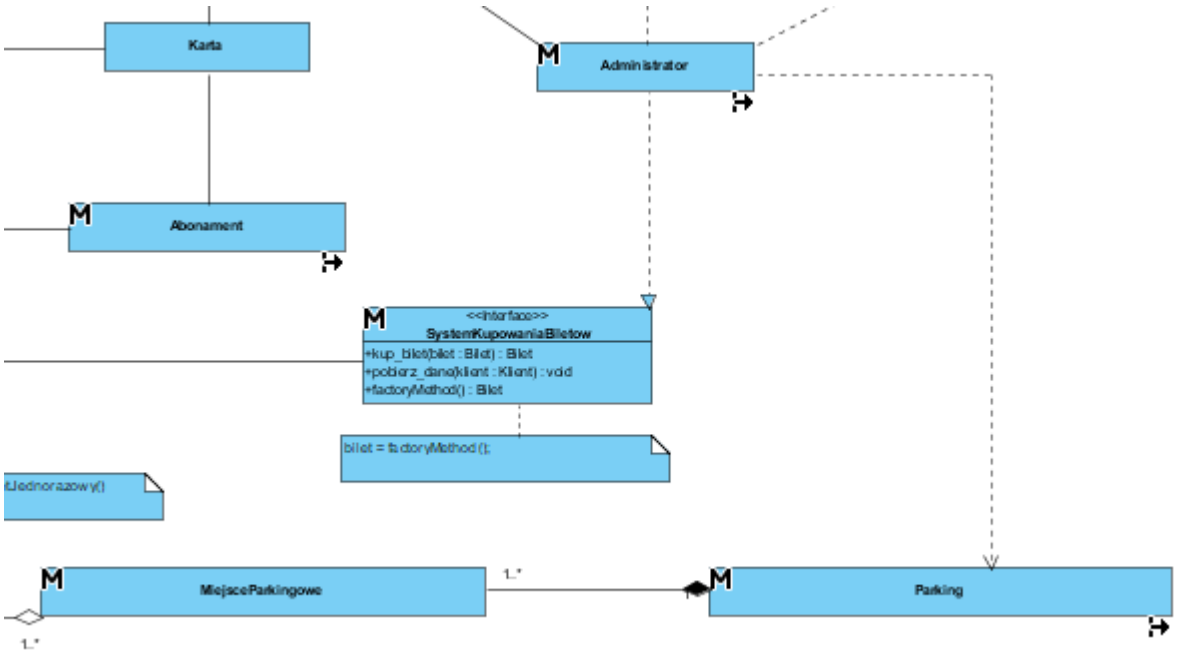


Klient komunikując się z „*SystememKupowaniaBiletów*" za pośrednictwem „*AutomatuBiletowego"* chce kupić pewien rodzaj biletu. Metoda „*FactoryMethod*()" wprowadza enkapsulację wiedzy na temat etapu samego powstawania biletu, dlatego też zwraca konkretny obiekt w zależności od rodzaju, jakiego potrzebował klient.

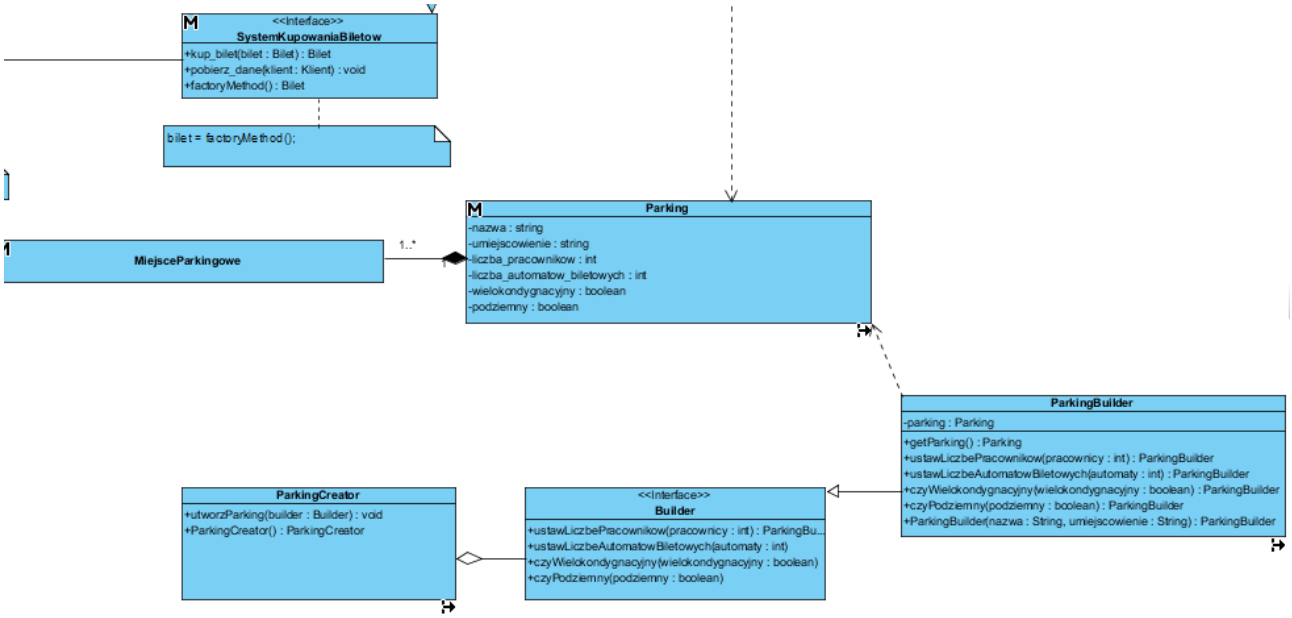
Wprowadzenie **metod wytwórczych** do tego projektu wydaje się obowiązkowe chociażby z logicznego punktu widzenia. Początkowa wersja projektu była zdecydowanie mniej czytelna, ponieważ proces tworzenia biletów był jawny dla klientów i nie było uwzględnionej żadnej określonej granicy, po której nie mają oni dostępu do procesu tworzenia biletów. Bilety jednorazowe oraz abonament w rzeczywistości dziedziczyły po klasie „*Bilet"*, a ich klasy były widoczne dla wszystkich. „AutomatBiletowy*"* przeładowywuje metody wytwórcze „*SystemuKupowaniaBiletów"*, zwracając instancje obiektów konkretnych biletów. Tworzenie obiektów wewnątrz klas **metod wytwórczych** jest bardziej elastyczne niż tworzenie obiektów wprost. Daje to możliwość rozszerzania wersji innych biletów w przyszłości.

Kolejnym zastosowanym wzorcem kreacyjnym w naszym projekcie jest budowniczy. Ze względu na mnogość konfiguracji przy tworzeniu parkingów. Wzorzec ten ze względu na swoją specyfikację zapewnia „płynne” tworzenie obiektów – stworzenie obiektu Parking jest prostsze (jednolinijkowy kod, który niweluje popełnienie błędu) oraz obsługuje sytuację, gdy nie chce podać się wszystkich atrybutów, które specyfikują tę klasę.

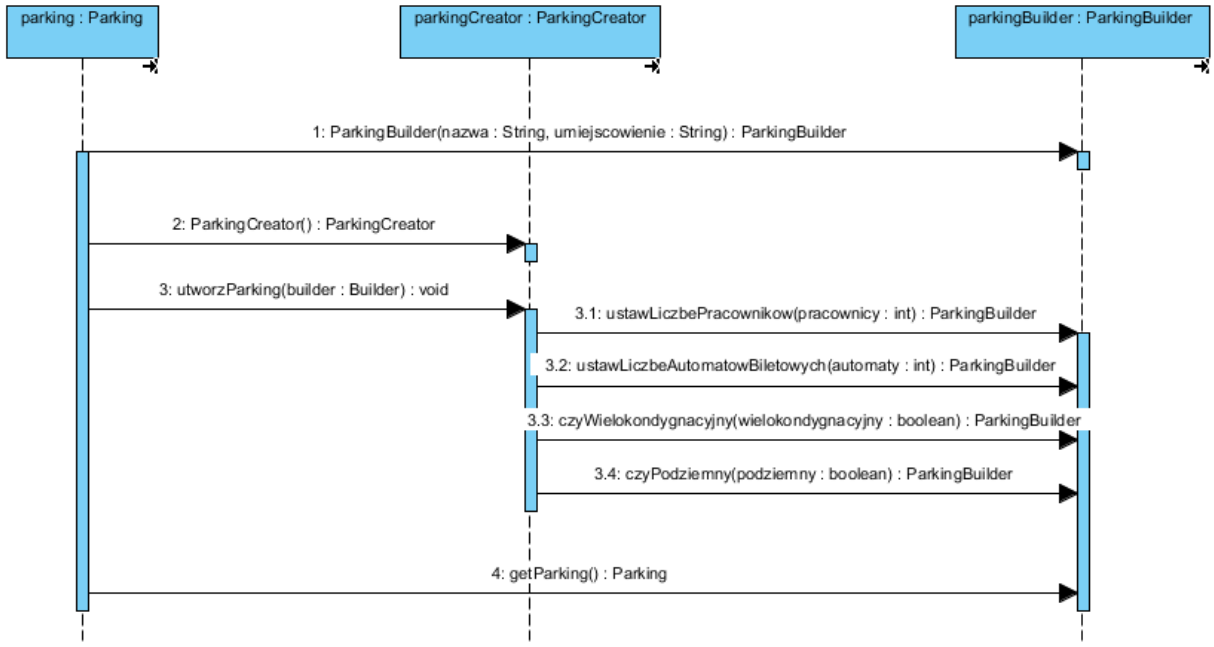
Fragment diagramu klas przed wykorzystaniem **budowniczego**:



Fragment diagramu klas po wykorzystaniu **budowniczego**:



Przebieg diagramu sekwencji dla budowniczego:



Chcąc stworzyć obiekt typu Parking tworzymy obiekt konkretnego budowniczego, który jako parametry przyjmuje obowiązkowe argumenty. Kolejnym krokiem jest stworzenie dyrektora oraz wywołanie na nim metody odpowiedzialnej za stworzenie parkingu (przyjmuje interfejs, który specyfikuje wymagania parkingu – Builder). Następnie twórca ustawia charakterystykę danego parkingu – nie musi wywołać wszystkich z podanych metod, a na samym końcu zwracamy zbudowany obiekt za pomocą operacji getParking(), który można przypisać do obiektu typu Parking. Ze względu na specyfikę tego wzorca projektowego nie wprowadza on żadnych modyfikacji do istniejącego projektu – jest on raczej dodatkiem, który usprawni pracę i zminimalizuje szansę pomyłki przy tworzeniu obiektu.