Programowanie obiektowe	Implementacja
Wypożyczalnia pojazdów – projekt laboratoryjny	
Niedziela, 17:15 **2019/2020**	**Kamil Zarych** **Dominik Cegiełka**

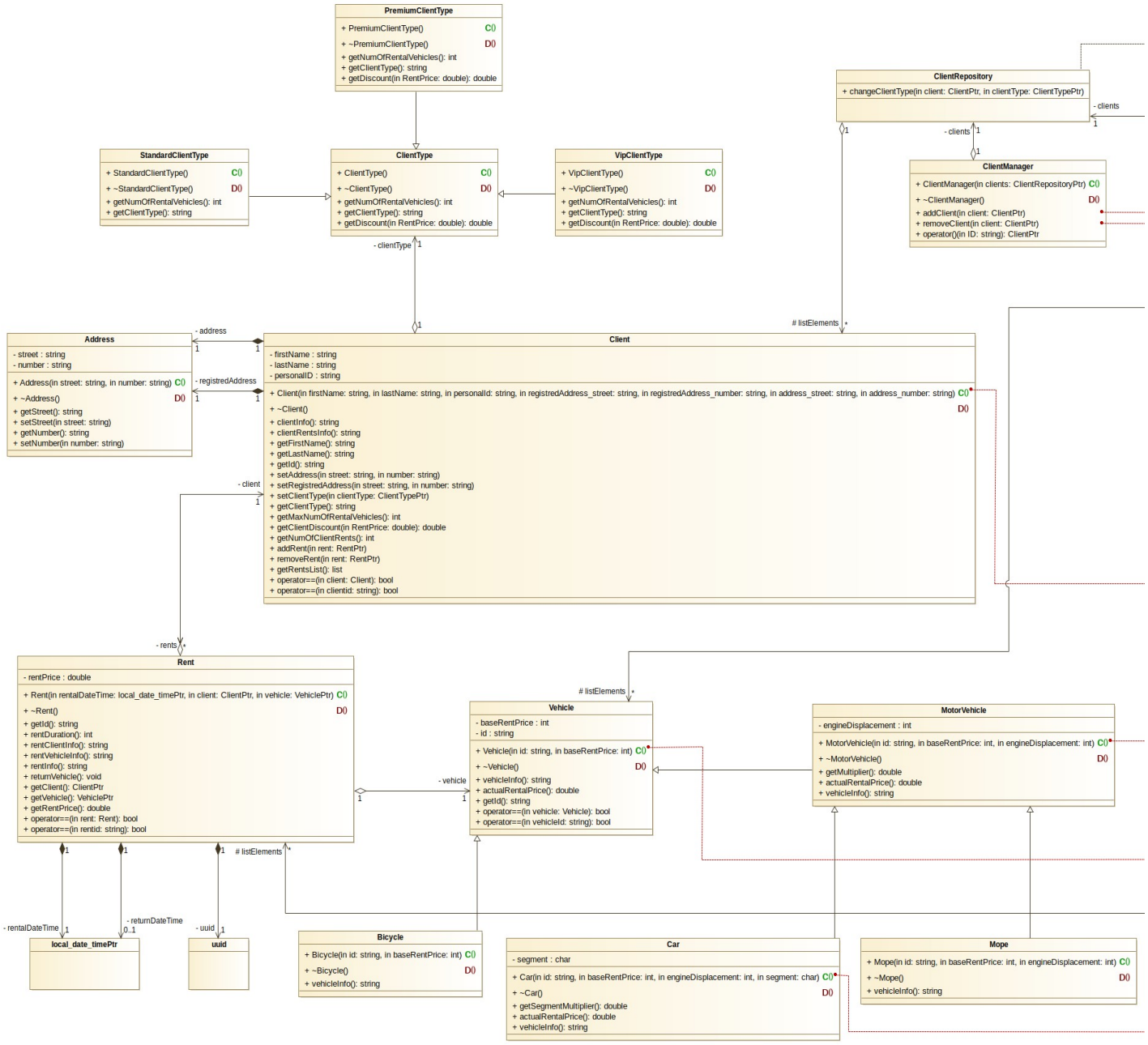
Opis funkcjonalności

Program pełni rolę wypożyczalni pojazdów. Klient ma do dyspozycji pojazdy różnego typu (samochód, motorower oraz rower), które może wypożyczać, a następnie zwracać. Cena bazowa za jeden dzień wypożyczenia jest stała (określona przy wpisaniu pojazdu do rejestru). Podczas tworzenia wypożyczenia ustalana jest aktualna cena za jeden dzień na podstawie parametrów danego typu pojazdu (segment, pojemność). Podczas zwracania pojazdu na podstawie łącznej wartości wypożyczenia obliczany jest upust w zależności od typu klienta. Typ klienta zależy od obrotu i jest ustalany po każdym zwrocie pojazdu. Program przechowuje kolekcje aktualnych oraz archiwalnych wypożyczeń.

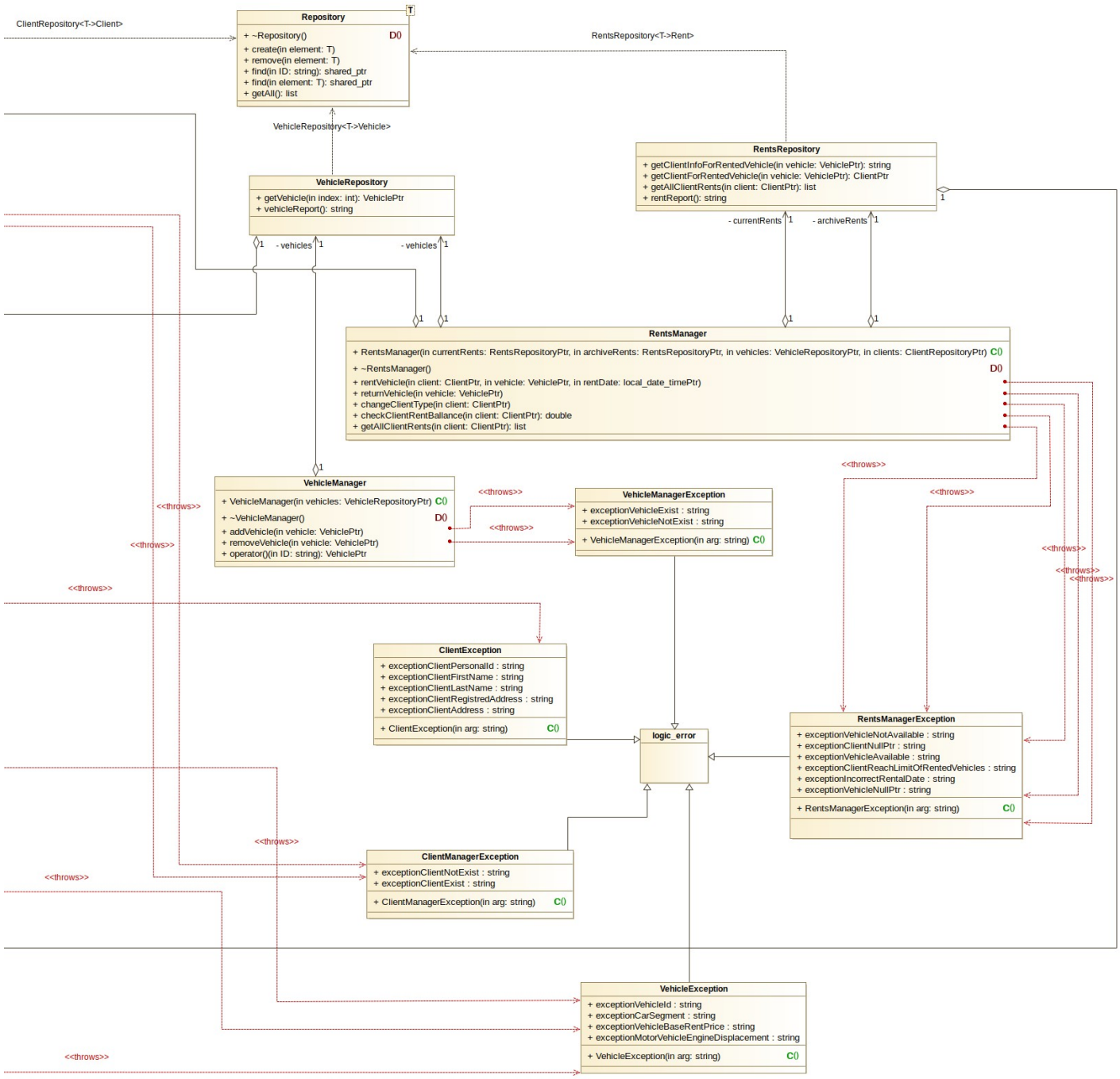
Zmiany

W programie zastosowaliśmy wyrażenie lambda oraz przeciążenia operatorów. Dodaliśmy wiele wyjątków w celu dokładniejszego sprawdzenia poprawności wprowadzanych danych. W klasie szablonowej Repository dodaliśmy metody szablonowe w celu uniknięcia powielania kodu w klasach pochodnych. Nie zastosowano mechanizmu utrwalania danych do pliku oraz graficznego interfejsu użytkownika.

Dokumentacja techniczna - diagram klas



Programowanie obiektowe	Implementacja



Address jest klasą definiującą adres zameldowania i adres zamieszkania klienta. Klasa ta jest powiązana relacją kompozycji z klasą **Client.**

Client jest klasą definiującą klienta wraz z jego danymi personalnymi. Klasa ta przechowuje aktualne wypożyczenia klienta na liście (dodaje i usuwa) oraz typ danego klienta (zmiana typu), zwraca informację o kliencie.

ClientType jest klasą abstrakcyjną, z której dziedziczą klasy poszczególnych typów klienta. Klasy pochodne wykorzystują zdefiniowany model.

StandardClientType , PremiumClientType, VipClientType są klasami definiującymi typy klientów, definiują maksymalną ilość jednocześnie wypożyczonych pojazdów oraz wyliczają należny upust.

Vehicle jest klasą definiującą podstawowy rodzaj pojazdu, dziedziczą po niej inne klasy pojazdów. Klasa przechowuje numer rejestracyjny pojazdu oraz bazową cenę. Zwraca informacje o pojeździe.

Bicycle jest klasą dziedziczącą z **Vehicle**, zwraca dane tego rodzaju pojazdu.

MotorVehicle jest klasą dziedziczącą z **Vehicle,** definiuje pojemność silnika na podstawie którego oblicza mnożnik ceny wypożyczenia.

Car jest klasą dziedziczącą z **MotorVehicle**, definiuje segment na podstawie którego oblicza mnożnik ceny wypożyczenia, stosuje wywołanie polimorficzne.

Mope jest klasą dziedziczącą z **MotorVehicle**, zwraca dane tego rodzaju pojazdu.

Rent jest klasą definiującą wypożyczenie pojazdu. Definiuje unikalny identyfikator, datę wypożyczenia i zwrotu pojazdu oraz cenę

wypożyczenia. Zawiera odniesienia do danych pojazdu i klienta.

Repository jest klasą szablonową definiującą listę elementów określonego typu. Wyznacza model dla klas pochodnych, posiada metody szablonowe (dodawanie, usuwanie, wyszukiwanie).

ClientRepository jest klasą dziedziczącą z klasy szablonowej **Repository**. Jest odpowiedzialna za zmianę typu klienta.

VehicleRepository jest klasą dziedziczącą z klasy szablonowej **Repository.** Zwraca raport pojazdów znajdujących się w repozytorium, zwraca pojazd na podstawie otrzymanego indeksu listy.

RentsRepository jest klasą dziedziczącą z klasy szablonowej **Repository.** Zwraca raport wypożyczeń znajdujących się w repozytorium, dane klienta, który wypożyczył określony pojazd oraz listę wszystkich wypożyczeń określonego klienta.

ClientManager jest klasą, która dodaje i usuwa klientów z repozytorium **ClientRepository**.

VehicleManager jest klasą, która dodaje i usuwa pojazdy z repozytorium **VehicleRepository.**

RentsManager jest klasą, która zarządza wypożyczeniami. Po zwrocie pojazdu przenosi wypożyczenie do listy wypożyczeń archiwalnych, sprawdza obrót klienta i na jego podstawie zmienia typ klienta, zwraca listę archiwalnych wypożyczeń klienta.

ClientException, ClientManagerException, RentsManagerException, VehicleException, VehicleManagerException są klasami wyjątków, definiują odpowiednie komunikaty.

Dokumentacja techniczna - wybrane scenariusze działania

- Pierwszy scenariusz

  Użytkownik tworzy klienta z poprawnymi danym (w przypadku błędnych danych zostanie rzucony wyjątek ClientException) a następnie dodaje go do repozytorium klientów ClientRepository przy pomocy managera ClientManager (metoda addClient(client)). Użytkownik ma możliwość zmiany adresów klienta, gdy zajdzie taka konieczność.

  Podczas dodawania i usuwania obiektu do/z repozytorium sprawdzane jest czy obiekt o tym samym identyfikatorze (PESEL, numer rejestracyjny, UUID) istnieje/nie istnieje w repozytorium i w przypadku niespełnienia wymaganego założenia rzucany jest odpowiedni wyjątek.

  Następnie użytkownik tworzy pojazd typu motorower przy pomocy klasy Mope. Pojazd zostaje dodany do repozytorium pojazdów VehicleRepository przy pomocy managera VehicleManager (metoda addVehicle(vehicle)).

  Następnie tworzone jest wypożyczenie przy pomocy klasy RentsManager, metoda rentVehicle(client, vahicle, rentDate). Podczas wypożyczania pojazdu sprawdzane jest czy klient istnieje i czy może wypożyczyć pojazd (limit wypożyczeń), czy wypożyczany pojazd istnieje i czy jest dostępny oraz czy data wypożyczenia nie jest z przyszłości. W przypadku nieprawidłowości zostanie rzucony wyjątek RentsManagerException. W przypadku gdy data wypożyczenia nie zostanie określona (nullptr), zostanie ustawiona aktualna data.

  Kiedy wszystkie przekazane dane są prawidłowe zostanie utworzone wypożyczenie z ustaloną kwotę wypożyczenia za jeden dzień na podstawie rodzaju pojazdu, a następnie dodane do repozytorium aktualnych wypożyczeń RentsRepository oraz do kolekcji wypożyczeń klienta.

- Drugi scenariusz

  Użytkownik zwraca wypożyczony pojazd za pomocą RentsManager, metodą returnVehicle(vehicle), jeśli podamy nieprawidłowy pojazd lub klient nie będzie posiadał wskazanego pojazdu zostanie rzucony wyjątek RentsManagerException.

  W przeciwnym przypadku metoda returnVehicle(vehicle) zwróci pojazd czyli usunie klientowi wypożyczenie, ustawi aktualną datę jako datę zwrotu pojazdu, obliczy łączny koszt wypożyczenia pojazdu uwzględniając upust klienta, przeniesie wypożyczenie z repozytorium aktualnych wypożyczeń do repozytorium wypożyczeń archiwalnych oraz zmieni typ klienta jeśli osiągnął wymagany dla danego typu obrót.

  Użytkownik ma możliwość wygenerowania raportu wypożyczeń bieżących jak i archiwalnych za pomocą RentsRepository, metodą rentReport() poprzez odpowiednie repozytorium currentRents lub archiveRents. Istnieje również możliwość wygenerowania raportu pojazdów z repozytorium VehicleRepository, metodą vehicleReport().

Dokumentacja techniczna - inne elementy nie opisane powyżej

Repozytoria zostały stworzone przy pomocy klasy szablonowej Repository.

Do tworzenia kolekcji w repozytoriach użyliśmy dynamicznej struktury z STL, szablon listy <list>.

Klient domyślnie jest typu Standard, który jest inicjalizowany w konstruktorze Klienta.

Pojazdy dziedziczą po klasie Vehicle i MotorVehicle. Dzięki zastosowaniu wywołań polimorficznych, w zależności od typu wywoływana jest metoda z odpowiedniej klasy nadrzędnej.

W programie wykorzystaliśmy inteligentne wskaźniki, które ułatwiają zarządzanie pamięcią i chronią przez wyciekiem pamięci. Można zastosować zwykłe wskaźniki, ale należy pamiętać o odpowiednim zwalnianiu pamięci (większe prawdopodobieństwo wycieku pamięci).

Do wyszukiwania obiektów w repozytoriach wykorzystaliśmy funkcję wyszukiwania find_if z STL z wyrażeniem lambda. Do sprawdzenia warunku wykorzystaliśmy przeciążenia operatorów w poszczególnych klasach (Client, Vehicle, Rent) i użyliśmy je w wyrażeniu lambda.

Dokumentacja użytkownika

Użytkownik może utworzyć klienta, podając wymagane dane (imię, nazwisko, pesel, adres zameldowania, adres zamieszkania) oraz dodać go do odpowiedniego repozytorium.

Użytkownik może utworzyć pojazd z wymaganymi danymi, zależnie od typu pojazdu (numer rejestracyjny, bazowa cena

Programowanie obiektowe	Implementacja

wypożyczenia, pojemność silnika, segment pojazdu) oraz dodać go do odpowiedniego repozytorium.

Użytkownik może utworzyć wypożyczenie podając klienta, pojazd oraz datę. W przypadku, gdy użytkownik nie poda daty, zostaje ona automatycznie ustalona na aktualną datę.

Użytkownik może zobaczyć listę wszystkich wypożyczeń danego klienta, sprawdzić który klient wypożycza dany pojazd, a także zobaczyć listę aktualnych oraz archiwalnych wypożyczeń.

Miejsce na uwagi prowadzącego