PROGRAMLAMA LABORATUVARI-I

II.PROJE RAPORU

1.Sümeyra Usta
Bilgisayar Mühendisliği(İ.Ö.)
Kocaeli Üniversitesi
Kocaeli/Türkiye
ust.sumeyra@gmail.com

I. ÖZET

Bu rapor Programlama Laboratuvarı I dersinin 2.projesi için oluşturulmuştur. Bu proje C++ dili kullanılarak geliştirilmiştir. Projeyi geliştirme ortamı olarak Qt Creator ve Qt Designer kullanılmıştır.

Belgede diyagramı , özet, giriş, yöntem ,deneysel sonuçlar gibi projeyi açıklayan başlıklara yer verilmiştir. Belge sonunda projenin sonucu ve projeyi hazırlarken kullanılan kaynaklar bulunmaktadır.

II. GİRİŞ

Bu rapor, C++ dilinde algoritma geliştirilen bir projenin detaylarını içermektedir. Projenin amacı admin ,company ,customer panelleri bulunan bilet almak için sefer eklenebilen/çıkarılabilen , şirket eklenebilen/çıkarılan ,araç eklenip/çıkarabilen bir arayüze bağlı OOP projesi geliştirilmesidir.

Projede ilk adımda admin paneli tanımlanmış, company ekleme çıkartma ve listeleme işlemleri yapılmıştır.

İkinci adımda ise company paneli yapılmıştır.Company panelinde araç ekleme/çıkartma , sefer ekleme/çıkartma ve listeleme yapılmıştır.

Üçüncü kısımda customer panel yapılmıştır.Bu panelde gidilmek istenen duraklar alınmış ve uygu seferin bulunup bulunmadığı yazılmıştır.

III. YÖNTEM

A. *Admin*

Admin sınıfının yapıcı fonksiyonu (Admin::Admin), beş adet önceden belirlenmiş şirketi (A, B, C, D, F) şirket listesine ekler. Bu işlem, ‘addCompany’ fonksiyonunu çağırarak gerçekleştirilir. Yapılandırıcı, temel sınıf ‘User'ın’ yapıcı fonksiyonunu da çağırarak, yönetici kullanıcının kimlik bilgilerini (kullanıcı adı ve şifre) ayarlar.

‘login’ metodunda, yönetici kullanıcının kullanıcı adı ve şifresi, sınıfın mevcut örnek değişkenleriyle karşılaştırılır. Eğer bilgiler eşleşirse, başarılı bir oturum açma (1 döndürülür) işareti verilir; aksi halde başarısızlık (0 döndürülür) bildirilir.

‘addCompany’ fonksiyonu, yeni şirketleri şirket listesine ekler. Her yeni şirket, ‘std::make_unique’ kullanılarak oluşturulan ve ‘std::unique_ptr<Company>’ ile yönetilen bir nesne olarak eklenir. Bu, bellek yönetimini otomatikleştirir ve bellek sızıntılarını önler. ‘removeCompany’ fonksiyonu, belirtilen isimdeki şirketi bulup listeden kaldırır. Bu işlem, ‘std::remove_if’ ve lambda ifadeleri kullanılarak gerçekleştirilir.

‘getCompanyList’ fonksiyonu, yöneticinin şirket listesine salt okunur erişim sağlar. Bu, şirket listesinin dışarıdan değiştirilmesini önler ve veri bütünlüğünü korur.

B. *Admin Paneli*

Bu kod parçası, ‘Qt framework’ kullanılarak geliştirilen bir grafik kullanıcı arayüzü (GUI)

uygulaması için adminpanel sınıfını tanımlar. Bu sınıf, bir yönetici kullanıcının arayüzünü temsil eder ve ‘QWidget’ sınıfından türetilmiştir.

Sınıf, bir yönetici nesnesi (Admin türünde) ve Qt'nin kullanıcı arayüzü elemanlarını yönetmek için ‘Ui::adminpanel’ nesnesini içerir. Yapıcı fonksiyon (adminpanel), bir Admin nesnesini parametre olarak alır ve bu nesne, sınıf içerisinde yönetici işlevlerini gerçekleştirmek için kullanılır. Yıkıcı fonksiyon ‘(~adminpanel)’, nesne yok edildiğinde gerekli temizlik işlemlerini gerçekleştirir.

Sınıfın içinde tanımlanan slotlar ‘(on_add_clicked, updateCompanyList, on_remove_clicked),’ kullanıcı arayüzündeki çeşitli etkileşimlere yanıt vermek için kullanılır. Örneğin, ‘on_add_clicked’ ve ‘on_remove_clicked’ slotları, kullanıcı arayüzündeki ilgili butonlara tıklandığında tetiklenir ve sırasıyla şirket eklemek veya şirketi kaldırmak için Admin nesnesinin yöntemlerini çağırır. ‘updateCompanyList’ slotu, arayüzdeki şirket listesini güncellemek için kullanılır.

‘companyListUpdated’ sinyali, şirket listesi güncellendiğinde diğer sınıflar veya ‘widget'lar’ tarafından kullanılabilir.

Bu sınıf, Admin sınıfının işlevselliğini grafiksel bir arayüzle birleştirir ve yönetici kullanıcılara şirketleri eklemek, kaldırmak ve güncel listeyi görüntülemek gibi işlemleri gerçekleştirmek için bir arayüz sağlar. Bu, kullanıcıların programın işlevlerine grafiksel bir arayüz üzerinden etkileşimli bir şekilde erişmesini kolaylaştırır.

C. Company

Bu kod parçası, bir taşıma şirketini temsil eden Company sınıfını tanımlar. Company sınıfı, User sınıfından türetilmiştir ve otobüs, uçak ve tren gibi çeşitli taşıtları yönetmek için tasarlanmıştır.

Sınıfın özellikleri arasında ‘name’ (şirketin adı), ‘buses’ (otobüs listesi), ‘airplanes’ (uçak listesi) ve ‘trains’ (tren listesi) bulunur. Bu listeler, sırasıyla ‘Bus*’, ‘Airplane*’ ve ‘Train*’ türünden pointerlar içerir. Bu tasarım, taşıtların polimorfik bir şekilde yönetilmesine olanak tanır.

Company sınıfının yapıcı fonksiyonu ‘(Company)’, şirket adını ve şifresini alır ve temel sınıfın ‘(User)’ yapıcı fonksiyonunu çağırır. Yıkıcı fonksiyon ‘(~Company)’, nesne yok edildiğinde gerekli temizlik işlemlerini gerçekleştirir.

Taşıtları eklemek ve kaldırmak için bir dizi fonksiyon tanımlanmıştır. ‘addBus’, ‘addAirplane’, ve ‘addTrain’ fonksiyonları sırasıyla bir otobüs, uçak veya tren eklemek için kullanılır. ‘removeBus’, ‘removeAirplane’, ve ‘removeTrain’ fonksiyonları ise belirli bir taşıtı ilgili listelerden kaldırır.

Her taşıt türü için ‘getBuses’, ‘getAirplanes’, ve ‘getTrains’ fonksiyonları, ilgili taşıt listesine salt okunur erişim sağlar. Bu, taşıt listelerinin dışarıdan değiştirilmesini önler ve veri bütünlüğünü korur.

‘getName’ fonksiyonu, şirket adına erişim sağlar.

Bu sınıf, bir taşıma şirketinin yönetimini simüle etmek için kullanılır. Otobüsler, uçaklar ve trenler gibi farklı taşıt türlerini yönetme yeteneği, bu sınıfı kapsamlı bir taşıt yönetim sistemi için uygun bir aday yapar. Sınıfın tasarımı, taşıtların yönetimini ve takibini kolaylaştıran esnek ve genişletilebilir bir yapı sunar.

D. Company Panel

Bu kod parçası, bir taşıma şirketinin grafik kullanıcı arayüzünü (GUI) temsil eden ‘companypanel’ sınıfını tanımlar. Sınıf, Qt framework kullanılarak geliştirilmiş ve QWidget sınıfından türetilmiştir.

‘companypanel’ sınıfının temel amacı, bir Company nesnesinin grafiksel arayüzünü sağlamaktır. Yapıcı fonksiyon ‘(companypanel)’, bir Company nesnesini parametre olarak alır ve bu nesne, sınıf içerisinde şirketin yönetim işlevlerini gerçekleştirmek için kullanılır. Yıkıcı fonksiyon ‘(~companypanel)’, nesne yok edildiğinde gerekli temizlik işlemlerini gerçekleştirir.

Sınıf, şirketin adını döndüren bir ‘getCompanyName’ fonksiyonu ve Company nesnesine erişimi sağlayan bir ‘getCompany’ fonksiyonu içerir.

Arayüzdeki çeşitli butonlara basıldığında tetiklenen bir dizi slot fonksiyonu tanımlanmıştır

‘(on_pushButton_clicked, updateBusList, updateTrainList, on_pushButton_2_clicked, updateAirplaneList, on_pushButton_3_clicked, on_pushButton_4_clicked, on_pushButton_5_clicked, on_pushButton_6_clicked)’. Bu slotlar, kullanıcı arayüzündeki etkileşimlere yanıt vermek için kullanılır ve sırasıyla otobüs, tren ve uçak listelerini günceller veya ilgili taşıtı ekler/kaldırır.

Bu sınıf, bir taşıma şirketinin taşıtlarını yönetmek için bir grafiksel arayüz sağlar ve kullanıcılara şirketin taşıt filosunu yönetme ve gözden geçirme yeteneği verir. Bu, kullanıcıların programın işlevlerine grafiksel bir arayüz üzerinden etkileşimli bir şekilde erişmesini kolaylaştırır.

E. User

Bu kod parçası, bir kullanıcıyı temsil eden User sınıfını tanımlar. User sınıfı, ‘ILoginable’ arayüzünden türetilmiştir ve temel kullanıcı bilgilerini (kullanıcı adı ve şifre) içerir.

Sınıfın özellikleri arasında ‘username’ (kullanıcı adı) ve ‘password’ (şifre) bulunur. Bu bilgiler, sınıfın korumalı üyeleri olarak tanımlanmıştır, böylece türetilen sınıflar tarafından erişilebilir ancak dışarıdan doğrudan erişilemez.

User sınıfının yapıcı fonksiyonu (User), kullanıcı adını ve şifresini alır ve bu değerleri sınıfın üye değişkenlerine atar.

‘Login’ fonksiyonu, ‘ILoginable’ arayüzünden geçersiz kılındığı için (override anahtar kelimesi ile) bu sınıfın bir örneğinin oturum açma işlemini gerçekleştirmesini sağlar. Bu fonksiyon, verilen kullanıcı adı ve şifrenin sınıfın üye değişkenleriyle eşleşip eşleşmediğini kontrol eder ve buna göre bir sonuç döndürür.

Bu sınıf, kullanıcıların temel özelliklerini ve oturum açma işlevselliğini temsil eder ve genellikle daha geniş bir kullanıcı yönetim sisteminin bir parçası olarak kullanılır. Bu sınıfın tasarımı, kullanıcı yönetimi ve doğrulamasını kolaylaştıran esnek ve genişletilebilir bir yapı sunar.

F. ILoginable

Bu kod parçası, ‘ILoginable’ adında bir arayüz sınıfını tanımlar. Bu arayüz, oturum açma işlevselliğini sağlayan bir sözleşme olarak hizmet eder ve bu işlevselliği uygulayan herhangi bir sınıf tarafından kullanılabilir.

‘ILoginable’ arayüzünün en önemli özelliği, saf sanal (pure virtual) bir login fonksiyonudur. Bu fonksiyonun imzası’ login(const std::string& username, const std::string& password)’ şeklindedir ve bu, uygulayan sınıfların bir kullanıcı adı ve şifre alarak oturum açma işlemini nasıl gerçekleştireceğini belirtir. Saf sanal olması, bu fonksiyonun ‘ILoginable'ı’ uygulayan her sınıf tarafından tanımlanması gerektiği anlamına gelir.

Ayrıca, ‘ILoginable’ sınıfı bir sanal yıkıcıya ‘(virtual ~ILoginable())’ sahiptir. Bu, arayüzü uygulayan sınıfların nesnelerinin doğru bir şekilde yok edilmesini sağlar ve bellek sızıntılarını önler.

Bu arayüz, farklı kullanıcı türleri için oturum açma işlevselliğini standartlaştırır ve farklı sınıfların ortak bir arayüz altında bu işlevselliği uygulamasına olanak tanır. Bu, nesne yönelimli programlamada polimorfizmi destekleyen ve kod tekrarını azaltan bir yaklaşımdır.

G. Vehicle

Bu kod parçası, ‘Vehicle’ adında bir arayüz sınıfını tanımlar. Bu sınıf, çeşitli taşıtların temel özelliklerini ve işlevselliğini tanımlar ve bu işlevselliği uygulayan herhangi bir taşıt sınıfı tarafından genişletilebilir.

‘Vehicle’ sınıfının özellikleri arasında name (taşıtın adı) bulunur. Bu bilgi, sınıfın özel üyesi olarak tanımlanmıştır ve dışarıdan doğrudan erişilemez, ancak ‘GetName’ fonksiyonu aracılığıyla okunabilir.

‘Vehicle’ sınıfının yapıcı fonksiyonu (Vehicle), taşıtın adını alır ve bu değeri sınıfın üye değişkenine atar.

Sınıfın iki önemli saf sanal fonksiyonu vardır: ‘GetNumberOfSeats’ ve ‘CalculateFuelCost. GetNumberOfSeats’ fonksiyonu, taşıtın koltuk sayısını döndürmek için kullanılır ve

‘CalculateFuelCost’ fonksiyonu, belirli bir mesafe için yakıt maliyetini hesaplamak için kullanılır. Her iki fonksiyon da ‘Vehicle'ı’ genişleten sınıflar tarafından tanımlanması gereken saf sanal fonksiyonlardır.

Bu arayüz, farklı taşıt türleri için temel özellikleri ve işlevleri standartlaştırır ve farklı taşıt sınıflarının ortak bir arayüz altında bu işlevleri uygulamasına olanak tanır. Bu, nesne yönelimli programlamada polimorfizmi destekleyen ve kod tekrarını azaltan bir yaklaşımdır.

H. Bus

Bu kod parçası, Bus adında bir sınıfı tanımlar. Bu sınıf, Vehicle sınıfından türetilmiştir ve bir otobüsü temsil eder.

Bus sınıfının özel üyeleri arasında ‘numberOfSeats’ (otobüsün koltuk sayısı) ve ‘routeNumber’ (otobüsün güzergah numarası) bulunur. Bu bilgiler, sınıfın yapıcı fonksiyonu (Bus) aracılığıyla alınır ve üye değişkenlere atanır.

Bus sınıfı, Vehicle sınıfından gelen iki saf sanal fonksiyonu ‘(GetNumberOfSeats ve CalculateFuelCost)’ geçersiz kılar (override). ‘GetNumberOfSeats’ fonksiyonu, otobüsün koltuk sayısını döndürür. ‘CalculateFuelCost’ fonksiyonu ise, belirli bir mesafe için yakıt maliyetini hesaplar. Bu hesaplama, otobüsün özel özelliklerine ve kullanım koşullarına göre uyarlanabilir.

Ek olarak, ‘Bus’ sınıfı ‘GetRouteNumber’ fonksiyonunu içerir. Bu fonksiyon, otobüsün ‘Route’ numarasını döndürmek için kullanılır.

Bu sınıf, bir otobüsün temel özelliklerini ve işlevlerini modelleyerek, geniş bir taşıt yönetim sistemi içerisinde kullanılabilecek bir yapı sunar. Bus sınıfı, polimorfik davranışları ve genişletilebilir özellikleri ile otobüslerin yönetimini ve takibini kolaylaştıran esnek bir yapıya sahiptir.

İ. Train

Bu kod parçası, Train adında bir sınıfı tanımlar. Bu sınıf, Vehicle sınıfından türetilmiştir ve bir treni temsil eder.

Bu bilgiler, sınıfın yapıcı fonksiyonu (Train) aracılığıyla alınır ve üye değişkenlere atanır.

Train sınıfı, Vehicle sınıfından gelen iki saf sanal fonksiyonu ‘(GetNumberOfSeats ve CalculateFuelCost)’ geçersiz kılar (override). ‘GetNumberOfSeats’ fonksiyonu, trenin toplam koltuk sayısını döndürür, bu sayı genellikle vagon sayısına ve her vagonun kapasitesine bağlıdır. ‘CalculateFuelCost’ fonksiyonu, belirli bir mesafe için yakıt (veya enerji) maliyetini hesaplar, bu hesaplama trenin elektrikli olup olmadığına ve diğer operasyonel faktörlere bağlı olabilir.

Bu sınıf, bir trenin temel özelliklerini ve işlevlerini modelleyerek, geniş bir taşıt yönetim sistemi içerisinde kullanılabilecek bir yapı sunar. Train sınıfı, polimorfik davranışları ve genişletilebilir özellikleri ile trenlerin yönetimini ve takibini kolaylaştıran esnek bir yapıya sahiptir.

J. Airplane

Bu kod parçası, Airplane adını taşıyan bir sınıf tanımlar ve bu sınıf Vehicle sınıfından türemiştir. Airplane sınıfı, bir uçağı temsil eder ve uçakların özgün özelliklerini ve işlevlerini içerir.

Sınıf, Vehicle sınıfından gelen ‘GetNumberOfSeats’ ve ‘CalculateFuelCost’ fonksiyonlarını geçersiz kılar. Bu fonksiyonlar, sırasıyla uçağın koltuk sayısını döndürür ve belirli bir mesafe için yakıt maliyetini hesaplar. Bu hesaplamalar, uçağın spesifik özelliklerine ve operasyonel koşullarına bağlı olarak gerçekleştirilir.

Bu sınıf, bir uçağın karakteristik özelliklerini ve işlevselliğini modelleyerek, geniş bir taşıt yönetim sistemine entegre edilebilecek esnek ve ayrıntılı bir yapı sunar.

K. Route

Bu kod parçası, Route adında bir sınıf tanımlar. Bu sınıf, bir ulaşım güzergahını ve bu güzergahtaki durakları temsil eder.

Route sınıfı, güzergah numarası (route_no), güzergahtaki durakların listesi (stops) ve duraklar arasındaki mesafeleri (distances) içeren özel üyelere sahiptir. Bu veriler, sınıfın yapıcı fonksiyonu aracılığıyla alınır.

Sınıf, iki ana işlev sunar: 'calculate_distance', başlangıç ve bitiş durakları arasındaki toplam mesafeyi hesaplar; 'has_stop', belirli bir durağın güzergahta olup olmadığını kontrol eder.

'find_stop_index' adında özel bir yardımcı fonksiyon da bulunur, bu fonksiyon bir durak adını alır ve bu durağın stops listesindeki indeksini bulur.

Route sınıfı, bir ulaşım güzergahını yönetmek ve bu güzergah üzerindeki mesafeleri hesaplamak için kullanılan pratik ve işlevsel bir yapı sunar.

L. Customer Panel

Bu kod parçası, bir müşteri panelini temsil eden 'customerpanel' adlı bir sınıfın uygulamasını içerir. Bu sınıf, Qt framework kullanılarak geliştirilmiş ve QWidget sınıfından türetilmiştir.

'customerpanel' sınıfı, kullanıcı arayüzünü oluşturmak için 'Ui::customerpanel' nesnesini kullanır. Sınıfın yapıcı fonksiyonu, arayüzü başlatır ve altı farklı sefer (sefer1'den sefer6'ya kadar) oluşturur. Bu seferler, Route sınıfı örnekleri olarak tanımlanır ve farklı duraklar ile mesafeler içerir.

'on_pushButton_clicked' fonksiyonu, bir butona tıklandığında tetiklenir. Bu fonksiyon, kullanıcı tarafından girilen kalkış ve varış noktalarını alır ve tanımlanan seferler içinde uygun bir seferin olup olmadığını kontrol eder. Eğer uygun bir sefer bulunursa, mesafe hesaplanır ve bir mesaj kutusu ile kullanıcıya bilgi verilir. Eğer uygun sefer bulunamazsa, kullanıcıya bir seferin bulunamadığına dair bilgi verilir.

Bu sınıf, kullanıcılara çeşitli güzergahlarda mevcut seferleri sorgulama ve bu seferler hakkında bilgi alma imkanı sunar. Bu işlevsellik, müşteri paneli arayüzü üzerinden etkileşimli bir şekilde sağlanır.

M. Main

Bu kod parçası, Qt framework kullanılarak geliştirilmiş bir ana pencere (MainWindow) sınıfının uygulamasını içerir. MainWindow sınıfı, QMainWindow sınıfından türetilmiştir ve uygulamanın ana arayüzünü temsil eder.

Ana pencerede, bir yönetici paneli (adminPanel), birden fazla şirket paneli (companyPanelList), ve bir müşteri paneli (cuspanel) oluşturulur. Yapıcı fonksiyon (MainWindow), arayüzü başlatır, bir Admin nesnesi oluşturur ve yönetici ve şirket panellerini kurar. Ayrıca, adminPanel'in 'companyListUpdated' sinyalini MainWindow'ın 'onCompanyListUpdated' slotuna bağlar.

'on_admin_clicked' fonksiyonu, yönetici giriş butonuna tıklandığında çalışır ve yönetici bilgilerinin doğruluğunu kontrol eder. Doğruysa, yönetici panelini gösterir.

'on_companyLog_clicked' fonksiyonu, bir şirketin giriş bilgilerini kontrol eder ve doğruysa ilgili şirket panelini gösterir.

'on_customer_clicked' fonksiyonu, müşteri butonuna tıklandığında çalışır ve müşteri panelini gösterir.

MainWindow sınıfının yıkıcı fonksiyonu (~MainWindow), oluşturulan nesneleri temizler.

Bu sınıf, uygulamanın ana kontrol noktasıdır ve kullanıcıların yönetici, şirket veya müşteri olarak farklı işlevselliklere erişimini sağlar. Bu işlevsellikler, kullanıcıların ihtiyaçlarına göre ayrı ayrı paneller üzerinden sunulur.

IV. SONUÇ

Sonuç olarak proje kapsamında OOP ve arayüz kullanımı , algoritmik düşünme ve isterleri birleştirme gibi kazanımlar öğrenilmiştir. Kullanıcıya admin ,company ve customer seçenekleri sunularak bir bilet satış platformu yazılmıştır.

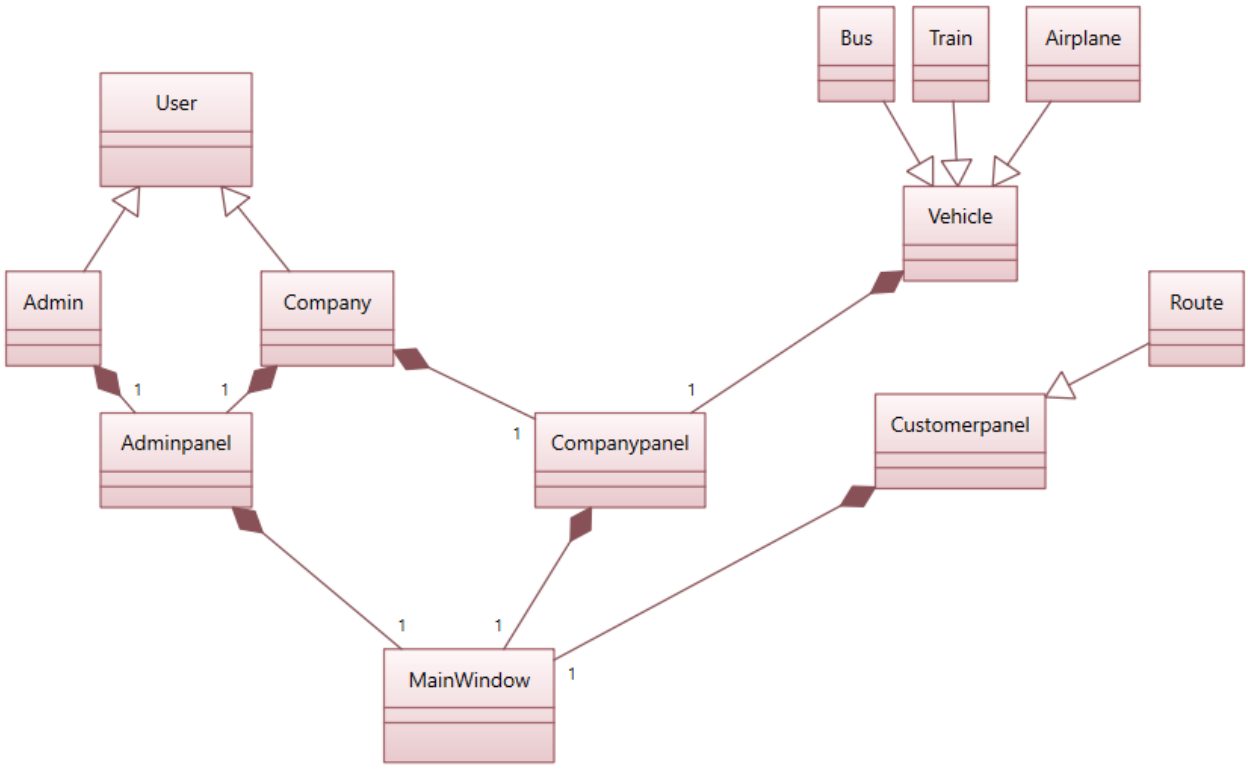
KAYNAKLAR

[1] https://www.youtube.com/watch?v=zP74WQNRoGg&list=PLOl6SW8nLgJzyGthQmsSikrsgspkym-ig

[2] https://wiki.qt.io/Qt_for_Beginners

[3] https://www.w3schools.com/cpp/cpp_oop.asp

[4] https://www.udemy.com/course/cpp_programlama/learn/lecture/33354832?start=0



Ek-1
UML DİYAGRAMI