

**KOCAELİ ÜNİVERSİTESİ**

**MÜHENDİSLİK FAKÜLTESİ**

**BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**

**2018**

**PROGRAMLAMA LABORATUVARI II (BLM210)**

**PROJE 4 RAPORU**

**ARAÇ ALIŞVERİŞ VERİTABANI PROGRAMI**

MEHMET OĞUZ AKTAŞ

150201170

oguzaktass@gmail.com

EBUBEKİR ŞİT

150201245

[sitebubekir@gmail.com](mailto:sitebubekir@gmail.com)

**İçindekiler**

Problem Analizi ve Çözüm Algoritmasının Oluşturulması 2

Yazılım Mimarisi 5

Akış Şeması7

Veritabanı ER Diyagramı 7

Yazılımın Çalıştırılması ve Özellikler 7

Kaynakça 8

Kazanımlar9

1 - Problem Analizi ve Çözüm Algoritmasının Oluşturulması

Bu projede, oluşturulan bir otomobil alışveriş veritabanının, MySQL veritabanı yönetim sistemi ve Java kullanılarak işlenmesi (kayıt ekleme, silme, güncelleme ve filtreleme temel işlemlerinin yapılması) istenmektedir. Problemin çözümünü daha kolay yapabilmek için problemi şu şekilde alt parçalara ayırdık;

1 - İstenen özelliklerde bir otomobil alışveriş veritabanının oluşturulması

2 - MySQL veritabanı ile Java arasında JDBC (Java Database Connectivity) API’i kullanarak bağlantı kurulması

3 - İstenen modüller için GUI tasarımının yapılması ve SQL sorgularının denenmesi

4 - 1. Modül için classların oluşturulup gerekli bileşenlerin yazılması

5 - 2. Modül için classların oluşturulup gerekli bileşenlerin yazılması

6 - Veritabanının ER (Entity-Relationship) diyagramının oluşturulması

7 - Veritabanı hatalarının düzeltilmesi

1 - İstenen özelliklerde bir otomobil alışveriş veritabanının oluşturulması

Veritabanını oluşturmak ve yönetmek için MySQL Workbench ve Navicat Premium Essentials for MySQL programlarını kullandık. İlk olarak MySQL Workbench üzerinden yeni schema oluşturup istenen 6 tabloyu (Tbl\_Araba, Tbl\_Ilan, Tbl\_Renk, Tbl\_VitesTuru, Tbl\_YakitTuru) ve tablolardaki kolonları oluşturduk. Daha sonra yine MySQL Workbench üzerinden veritabanının EER diyagramını oluşturarak tablolar arasındaki bağlantıları (foreign keyleri) diyagram üzerinden tanımladık. Son olarak veritabanına uygun kayıtlar eklemek için <https://www.sahibinden.com/kategori/otomobil> adresindeki ilanlardan yararlanarak, INSERT INTO ifadesiyle Tbl\_Ilan ve Tbl\_Araba tablolarına 34 kayıt, Tbl\_Renk ve Tbl\_Sehir tablolarına 11 kayıt, Tbl\_YakitTuru tablosuna 4 kayıt, Tbl\_VitesTuru tablosuna 3 kayıt eklenmesini sağlayacak olan “insert-records.sql” dosyasını hazırlayıp MySQL Workbench’e bu dosyayı import edip çalıştırdık. Ayrıca tabloların yanlışlıkla silinme veya çözemeyeceğimiz bir hata çıkması olasılığına karşı, veritabanını ve tabloları oluşturmak için gerekli SQL ifadelerinin bulunduğu “create-db-tables.sql” dosyasını hazırladık.

2 - MySQL veritabanı ile Java arasında JDBC (Java Database Connectivity) API’i kullanarak bağlantı kurulması

Bu işlemi yapabilmek için NetBeans IDE’de oluşturduğumuz proje özelliklerinden, içinde MySQL Connector J 5.1.23 veritabanı driverını bulunduran MySQL JDBC Driver kütüphanesini ekledik. Daha sonra NetBeans IDE’de Services bölümündeki Databases sekmesinde bulunan Drivers’tan MySQL Connector J’I seçerek “Connect using” seçeneği ile host (localhost), port (3306), veritabanı (arac\_alisveris), kullanıcı adı (root), şifre (123456) alanlarını doldurarak JDBC URL bölümüne “jdbc:mysql://localhost:3306/arac\_alisveris” yazarak MySQL veritabanı ile bağlantı oluşturduk. Bağlantıyı sağladıktan sonra <https://docs.oracle.com/javase/tutorial/jdbc/basics/connecting.html> adresindeki örneğe göre java.sql API’I içerisinde bulunan DriverManager classını, DriverManager.getConnection() metodunu ve yine java.sql API’i içerisindeki Connection interface’ini kullanarak veritabanı ile bağlantı sağladık. SQL ifadelerini çalıştırabilmek için Statement interface’ini ve ondan extend edilen PreparedStatement interface’ini kullandık. SELECT sorgularını konsolda ve yazdığımız JFrame classlarında JTable tablolarında gösterebilmek için ResultSet nesnesini kullandık. Oluşturduğumuz tüm GUI classlarında (JFrame'den extend edilen classlarda) veritabanı ile bağlantı kuracağımızda DriverManager.getConnection() metodunu tekrar yazdık.

3 - İstenen modüller için GUI tasarımının yapılması ve SQL sorgularının denenmesi

1. modülde istenen Insert, Update, Delete işlemlerini yapabilmek için tüm tablolar için ayrı JFrame formları oluşturduk. Daha sonra NetBeans IDE’de Design kısmından istediğimiz tasarımı yaptık. Insert, Update ve Delete butonlarını hazırlayıp ActionPerformed() metotlarıyla butona basıldığında çalıştırılacak kodları (veritabanına bağlanıp gerekli SQL sorgularını çalıştırmayı) yazdık. Tablolarda bulunan kolonlar için gerekli alanları girmek için JTextField’lar oluşturduk. Ayrıca daha sonra kayıtları listelemek için JTable’lar oluşturup tablo üzerinde bir satıra tıklandığında bunun text fieldlarda ilgili alana yazılmasını sağlayacak showItem() metotlarını yazdık. Kullandığımız SQL sorguları 6 tablo için de aynı. Tablolardaki kayıtları görüntülemek için “SELECT \* FROM Tbl\_Ilan”, Update için “UPDATE Tbl\_Ilan SET Ilan\_Adi = ‘<yeni\_deger>’” şeklinde (burada <yeni\_deger> kısmı JTextField ile kullanıcıdan alınacak), Delete için “DELETE FROM Tbl\_Ilan WHERE IlanID = ‘<ID>’, Insert için “INSERT INTO Tbl\_Ilan(Ilan\_Adi, Ilan\_Fiyat, Ilan\_Km, Ilan\_Tarih, Ilan\_ArabaID, Ilan\_SehirID) VALUES (‘?’, ‘?’, ‘?’, ‘?’, ‘?’, ‘?’) şeklinde PreparedStatement interfaceini kullanarak SQL sorguları yazdık ve veritabanına bağlanıp çalıştırdık. Diğer 5 tabloyu da (Tbl\_Araba, Tbl\_YakitTuru, Tbl\_VitesTuru, Tbl\_Renk, Tbl\_Sehir) benzer şekilde yaptık. Tbl\_Ilan tablosunda diğerlerinden farklı olarak tarih kısmı da var, tarihi kullanıcıdan input olarak alabilmek için jcalendar-1.4 kütüphanesini projeye import ettik ve JDateChooser componentini kullandık. 2. modül için GUI tasarımını önce 7-8 sorgu için ayrı JFrame’ler oluşturarak yapmayı veya bir JFrame içinde ayrı JPanel'ler oluşturarak yapmayı planlamıştık fakat hepsini tek bir JFrame içinde tek bir JPanel'de yapabileceğimizi fark edince bu şekilde yaptık. 1. modüle benzer şekilde 2. modüldede (FiltrelemeWindow classında) istenen inputları almak için JTextField'lar, JComboBox'lar, JCheckBox'lar, Sorgula isminde bir JButton ve sorgu sonucunu göstermek için bir JTable kullandık.

4 - 1. Modül için classların oluşturulup gerekli bileşenlerin yazılması

1. modülde tüm tablolar için veritabanındaki kayıtları okuyup ArrayList tipinde kaydedebilmek için ayrı GUI dışında bir class daha oluşturduk. Örneğin Araba classında int tipinde id, vitesturuid, yakitturuid, renkid ve String tipinde marka ve model özellikleri ve bunların get-set metotları ve constructor metodu bulunuyor. ArabaWindow GUI classında getArabaList() metodu ilk olarak “SELECT \* FROM Tbl\_Araba” sorgusuyla Tbl\_Araba tablosundaki tüm kayıtları okur ve while döngüsü içinde bunları tek tek new Araba() constructor metoduna parametre olarak göndererek yeni Araba nesnesi oluşturur. Daha sonra ArrayList<Araba> tipinde arabalistesi ArrayList’ine ekler. Tüm tablolar için getList() metotlarını bu şekilde yazdıktan sonra oluşturduğumuz JTable’da göstermek için farklı bir metot daha yazdık. Örneğin ArabaWindow classı içinde arabalaritablodaGoster() metodu önce getArabaList() metodu ile veritabanındaki kayıtları alıp for döngüsü içinde bunları oluşturulan tablodaki sütunlara addRow() hazır metodunu kullanarak tek tek ekler. Bunun dışında showItem() metodu ile tabloda üzerine tıkladığımız bilgilerin gerekli text fieldlarda görünmesini sağladık, böylece Delete işlemi için gerekli alanları tek tek doldurmaya gerek kalmadı. Son olarak Tbl\_Ilan ve Tbl\_Araba tabloları için yazdığımız comboboxgetList() metotlarıyla tabloda üzerine tıkladığımız kayıtta ilgili alanın JComboBox'ta seçili olmasını sağladık.

5 - 2. Modül için classların oluşturulup gerekli bileşenlerin yazılması

2. modülde (FiltrelemeWindow classı içinde) yazdığımız alttaki sorgu ile veritabanına bağlanıp Tbl\_Ilan tablosundan IlanID, Ilan\_Adi, Ilan\_Fiyat, Ilan\_Km, Ilan\_Tarih kolonlarını, Tbl\_Araba tablosundan Araba\_Marka, Araba\_Model kolonlarını, Tbl\_YakitTuru tablosundan Yakit\_Turu kolonunu, Tbl\_VitesTuru tablosundan Vites\_Turu kolonunu, Tbl\_Renk tablosundan Renk kolonunu, Tbl\_Sehir tablosundan Sehir kolonunundaki kayıtları almayı sağladık.

*"SELECT i.IlanID, i.Ilan\_Adi, i.Ilan\_Fiyat, i.Ilan\_Km, i.Ilan\_Tarih, a.Araba\_Marka, a.Araba\_Model, y.Yakit\_Turu, v.Vites\_Turu, r.Renk, s.Sehir FROM Tbl\_Ilan i LEFT JOIN Tbl\_Araba a ON i.Ilan\_ArabaID = a.ArabaID LEFT JOIN Tbl\_YakitTuru y ON a.Araba\_YakitTuruID = y.YakitTuruID LEFT JOIN Tbl\_VitesTuru v ON a.Araba\_VitesTuruID = v.VitesTuruID LEFT JOIN Tbl\_Renk r ON a.Araba\_RenkID = r.RenkID LEFT JOIN Tbl\_Sehir s ON i.Ilan\_SehirID = s.SehirID WHERE 1=1";*

Daha sonra bu kayıtları oluşturduğumuz JTable’a atabilmek için içinde gerekli id, ilanadi, fiyat, yakitturu gibi 11 tane özelliği bulunduran SQL classını yazdık. FiltrelemeWindow classında ArrayList<SQL> tipindeki getList() metodu ile üstteki SQL sorgusunun sonuçlarını new SQL() constructorı ile yeni nesnelere atadık, daha sonra bu nesneleri tablodaGoster() metodunda kullanarak JTable’ın kolonlarına atadık. Ayrıca oluşturduğumuz JComboBox'ları da mevcut kayıtlarla doldurmak için getList() metodu içinde ilgili tabloda SELECT sorguları ile ilgili kolondaki kayıtları combobox’a ekledik.

Son olarak projenin en önemli kısmı olan filtreleme için, oluşturduğumuz Sorgula butonunun ActionPerformed() metodunda, üstteki SQL sorgusunun sonuna “query +=” kodlarıyla JTextField, JComboBox, JCheckBox alanlarıyla kullanıcıdan aldığımız inputlara göre “ AND r.Renk = ‘<renk>’” şeklinde eklemeler yaptık. Örneğin ilan adına göre azalan sıralama seçiliyse sorgunun sonuna “ ORDER BY i.Ilan\_Adi DESC” Stringi ekleniyor. Daha sonra filtreleme sonunda sorgunun son halini veritabanında çalıştırıp yine döngü içinde SQL classı tipinde records özelliğine kaydettik. Son olarak bu records’u döngü içinde oluşturduğumuz tablodaki kolonlara ekleyerek doğru bir şekilde filtreleme işlemini tamamladık.

6 - Veritabanının ER (Entity-Relationship) diyagramının oluşturulması

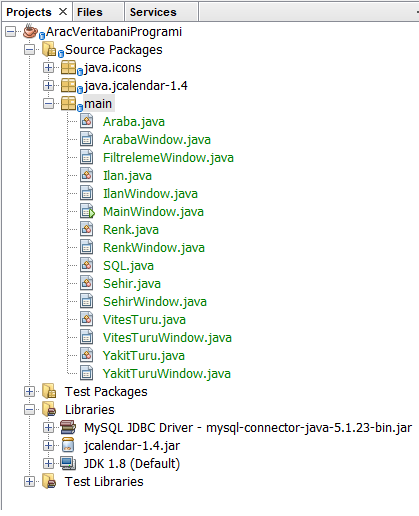
İlk aşamada veritabanını hazırlarken MySQL Workbench programında foreign keyleri tanımlamak için ER diyagramını oluşturup diyagram üzerinden tanımlamıştık. Projeyi bitirdikten sonra Navicat for MySQL programlarını kullanarak bir tane daha ER diyagramı oluşturduk. Bunlardan MySQL Workbench ile oluşturduğumuzu beğendiğimiz için onu kullandık.

7 - Veritabanı hatalarının düzeltilmesi

Programda denemeler yaparken bazı durumlarda Delete işleminin çalışmadığını fark ettik. Bunun nedeninin bir tablonun başka bir tabloya foreign key kısıta ile bağlı olmasını ve veritabanı oluştururken “creates-db-tables.sql” dosyasında foreign keyleri tanımlarken “ON DELETE NO ACTION” seçeneği ile tanımlamamızdan dolayı olduğunu anladık. Bu kısmı “ON DELETE CASCADE” yaptığımızda örneğin Tbl\_Araba tablosunda sileceğimiz kayıt Tbl\_Ilan tablosunda Ilan\_ArabaID sütununa bağlı kaydın da silinmesine sebep oluyordu. Bu nedenle foreign keyleri “ON DELETE SET NULL” seçeneğiyle tanımladık.

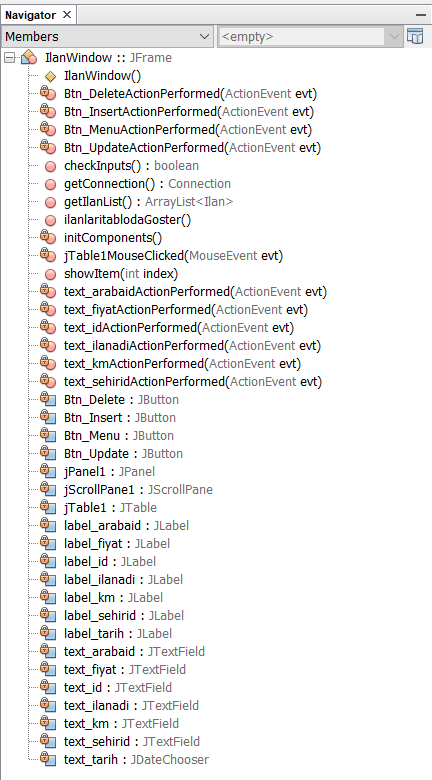
2 - Yazılım Mimarisi

Programı yazmak için JDK 8u172 sürümüyle NetBeans 8.2 IDE’yi, veritabanı bağlantısı için Java Database Connectivity kütüphanesi içinde bulunan MySQL Connector J 5.1.23’ü, GUI için Swing ve AWT API’lerini kullandık.



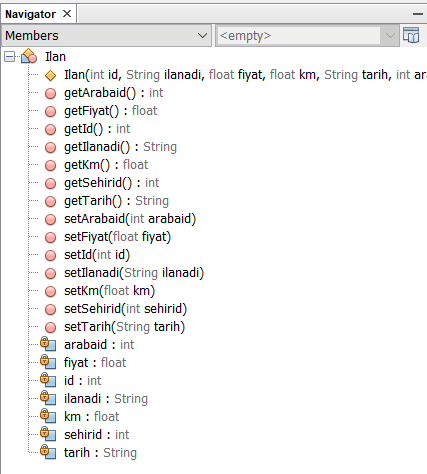
*Şekil 1: Programda bulunan paketler, sınıflar ve kütüphaneler*

Uygulamada MainWindow, IlanWindow, ArabaWindow, YakitTuruWindow, VitesTuruWindow, RenkWindow, SehirWindow ve FiltrelemeWindow isimlerinde 8 JFrame formu şeklinde GUI classı yazdık. Sorgu sonuçları tabloda gösterebilmek için ve 1. bölümde açıkladığımız diğer sebeplerden dolayı 6 tablo için Ilan, Araba, YakitTuru, VitesTuru, Renk, Sehir isimlerinde normal class ve filtreleme bölümünde tüm tablolardan bazı kolonların özelliklerinin bulunduğu SQL isminde bir class daha oluşturduk. Bu classları veritabanındaki kayıtları GUI’de oluşturduğumuz JTable’lara atabilmek için kullandık. Tüm classlar main kütüphanesinde bulunuyor. java.icons kütüphanesinde butonlar için gerekli .jpg biçiminde ikonlar bulunuyor. java.jcalendar-1.4 kütüphanesini de GUI’de JDateChooser componentini kullanabilmek için ekledik.



*Şekil 2: IlanWindow classı genel yapısı*

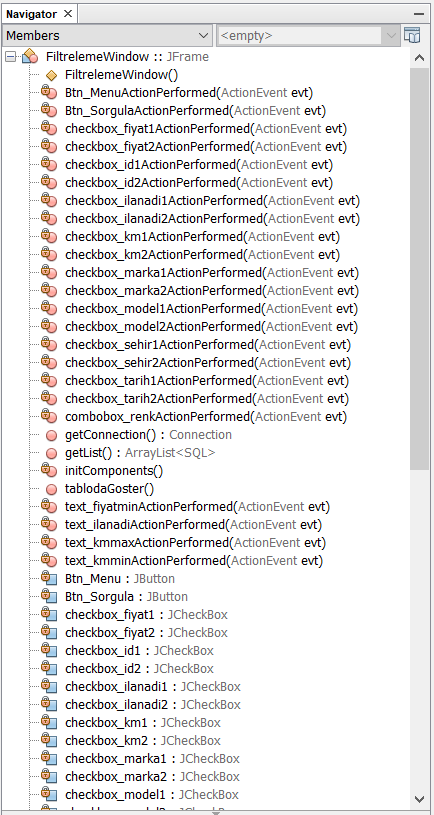
IlanWindow classında 1. modül için gerekli olan Insert, Update, Delete işlemleri için gerekli GUI componentleri ve üstte hangi amaçla kullandığımızı açıkladığımız Connection tipinde getConnection(), ArrayList<Ilan> tipinde getIlanList(), ilanlaritablodaGoster(), showItem() metotları bulunuyor. Bunların yanında text fieldler ile kullanıcıdan alınan değerleri kontrol etmek için checkInputs() metodunu yazdık. Örneğin ID kısmını boş bırakıp Update butonuna tıkladığımızda JOptionPane tipinde bir “Bir ya da daha fazla alani bos biraktiniz.” mesajının açılmasını sağladık. checkInputs() metodunda aynı zamanda fiyat ve km özelliklerinin alttaki Float tipine çevrilmesini sağladık.



*Şekil 3: Ilan classı genel yapısı*

Ilan classında Tbl\_Ilan tablosunda bulunan kolonlara benzer şekilde id, ilanadi, fiyat, km, tarih, arabaid, sehirid özelliklerini tanımlayıp constructor metodu ile gerekli get ve set metotlarını yazdık.

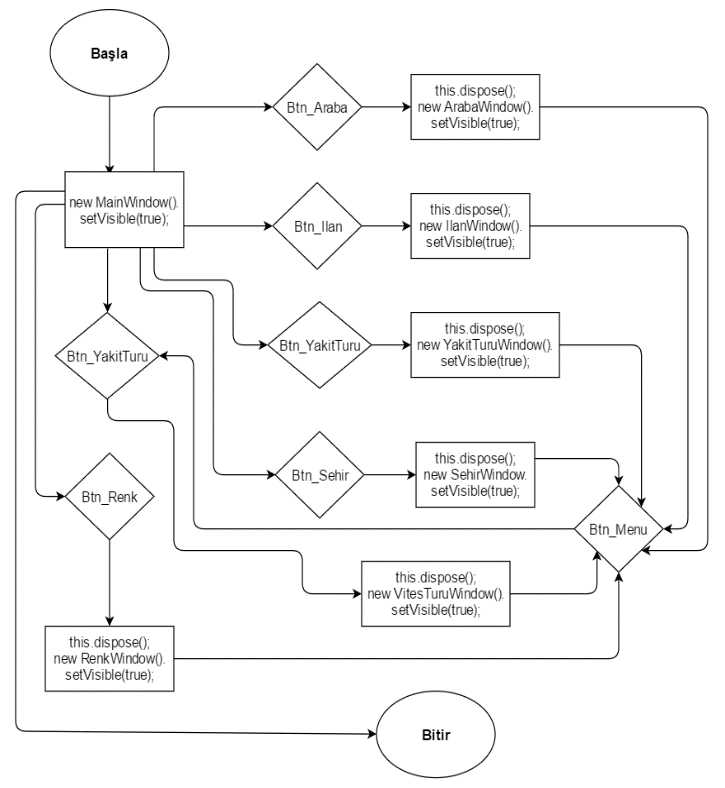
Diğer 5 tablo için de GUI classı ile normal classları tablolarda bulunan kolonlara uygun şekilde Tbl\_Ilan tablosu için yazdıklarımıza benzer şekilde yazdık.



*Şekil 4: FiltrelemeWindow classı genel yapısı*

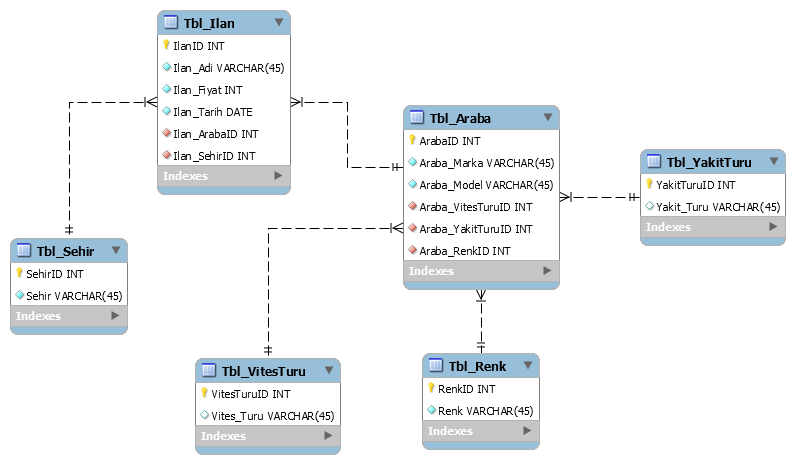
FiltrelemeWindow classında filtreleme işlemleri için kullanacığımız GUI bileşenlerini (labellar, butonlar, textfieldlar, comboboxlar, checkboxlar ve en önemlisi de yapacağımız sorgu sonuçlarını göstereceğimiz tabloyu) tasarladık. Oluşturduğumuz JTable’a veritabanındaki 6 tablodan seçtiğimiz ve bizden istenen 11 tane kolonu LEFT JOIN ifadeleriyle 1. bölümde yazdığımız SQL sorgusuyla ekledik. Daha sonra gerekli GUI bileşenlerini kullanarak ilk sorgu içinde WHERE kısmından sonra eklemeler yapıp filtreleme işlemini bizden istendiği şekilde yaptık.

3 - Akış Şeması



*Şekil 5: Akış şeması*

4 - Veritabanı ER Diyagramı



*Şekil 6: Veritabanı ER diyagramı*

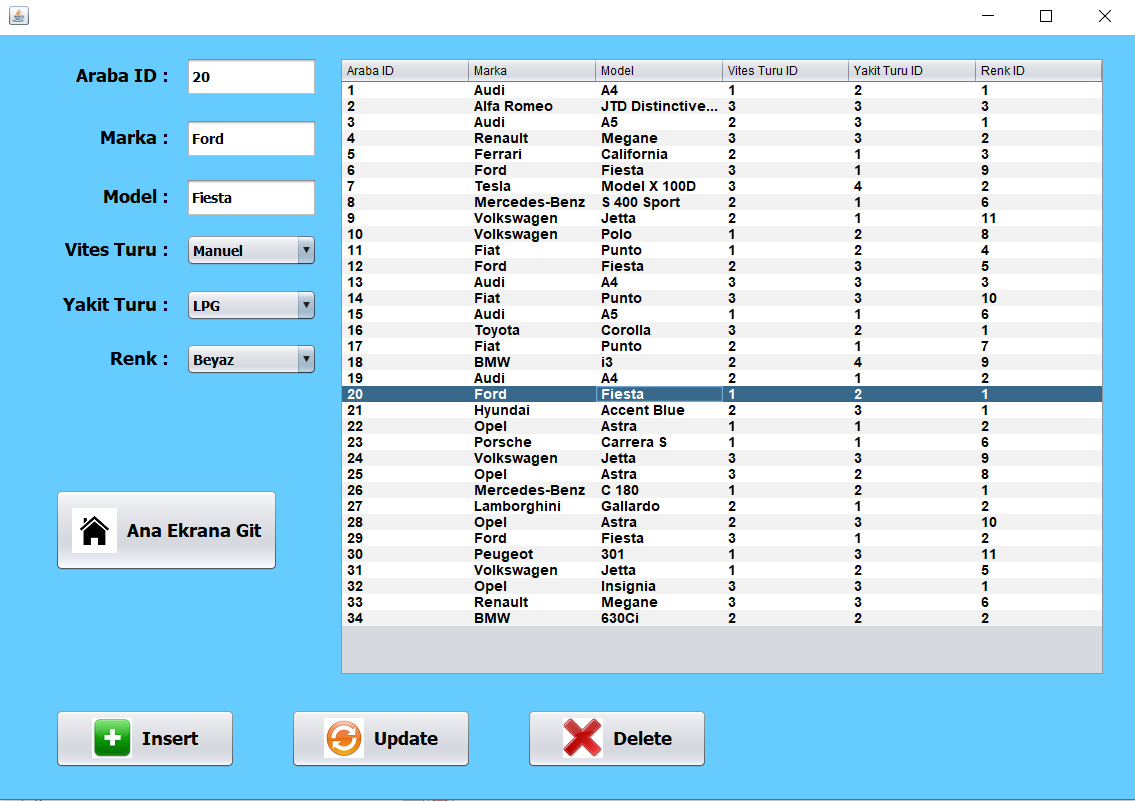
5 - Yazılımın Çalıştırılması ve Özellikleri

Programı çalıştırınca ilk olarak ana menü açılıyor.



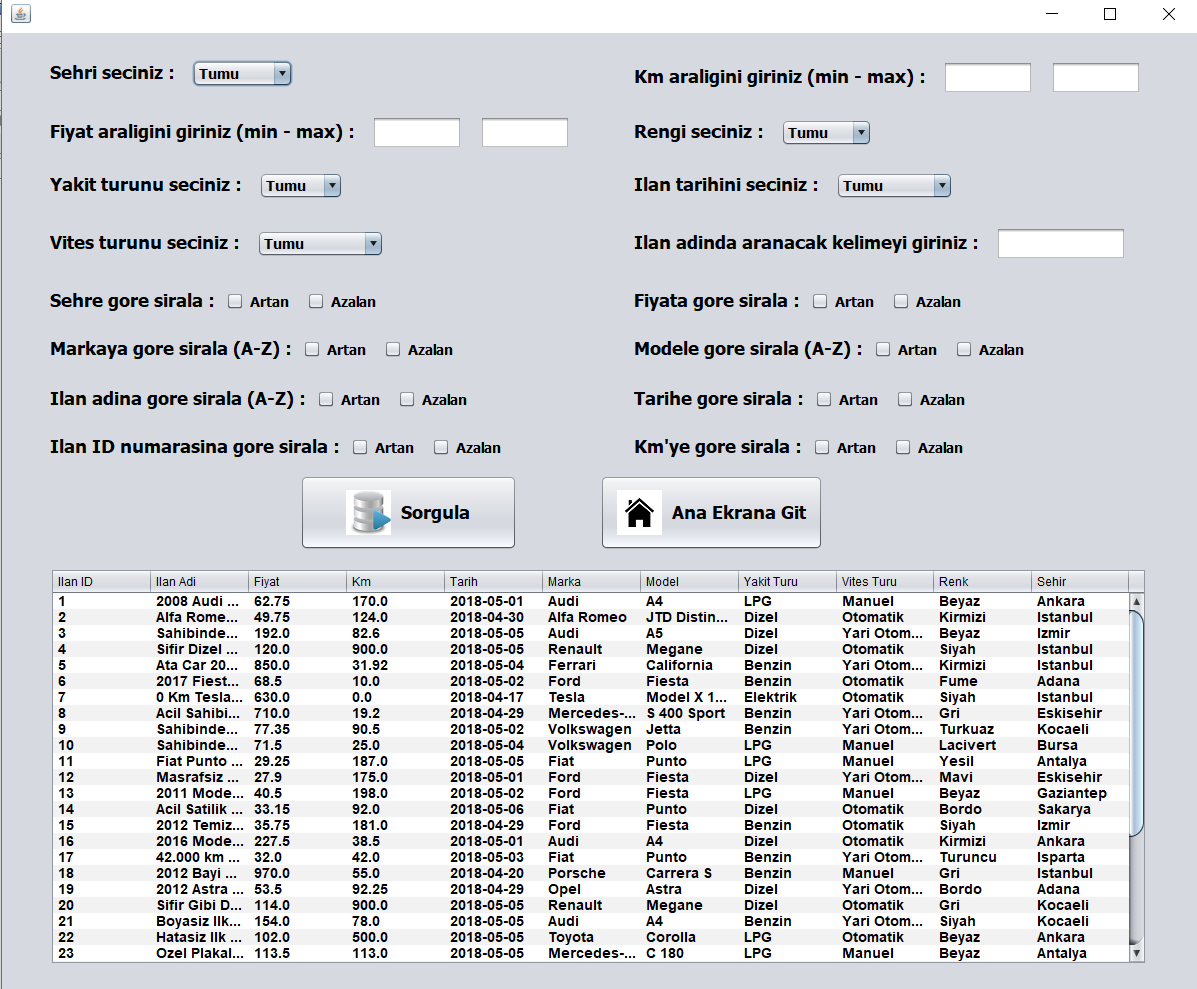
*Şekil 7: Programın ilk çalıştırılması, ana menü*

Buradan 1. modülde Insert, Update, Delete işlemlerinin yapılacağı tablo seçilebilir veya 2. modülde filtreleme işlemi seçilebilir.



*Şekil 8: Tbl\_Araba tablosu 1. modül işlemleri*

Ana ekrana git seçeneği seçilmediği sürece pencere kapatılamaz. Programın alt menüler için pencere kapatma seçeneğini devre dışı bıraktık, sadece ana menüden pencere kapatılıp program sonlandırılabilir.



*Şekil 9: 2. modül filtreleme işlemleri*

6 - Kaynakça

[1] <https://dev.mysql.com/doc/>

[2] <https://docs.oracle.com/javase/8/docs/api/>

[3] <https://docs.oracle.com/javase/tutorial/jdbc/basics/index.html>

[4] <https://docs.oracle.com/cd/E17952_01/mysql-5.7-en/mysql-5.7-en.pdf>

[5] <https://stackoverflow.com/>

[6] <https://www.quora.com/How-do-I-connect-Java-NetBeans-to-MySQL-using-JDBC> (Erişim tarihi: 05.05.2018)

[7] <http://www.mysqltutorial.org/jdbc-overview/> (Erişim tarihi: 05.05.2018)

[8] <https://dba.stackexchange.com/questions/1/what-are-the-main-differences-between-innodb-and-myisam> (Erişim tarihi: 05.05.2018)

[9] <https://www.youtube.com/playlist?list=PLh9ECzBB8tJNp5a5yjuuqL6jMbSXcTxHo> (Erişim tarihi: 06.05.2018)

[10] <https://www.youtube.com/watch?v=vtTUKLE_SWE> (Erişim tarihi: 06.05.2018)

[11] <https://www.tutorialspoint.com/jdbc/jdbc-db-connections.htm> (Erişim tarihi: 06.05.2018)

[12] <https://www.cs.mun.ca/java-api-1.5/guide/jdbc/getstart/resultset.html> (Erişim tarihi: 06.05.2018)

[13] <https://www.developer.com/java/creating-a-jdbc-gui-application.html> (Erişim tarihi: 06.05.2018)

[14] <http://creativetuts.blogspot.com.tr/2013/01/insert-update-delete-view-operations-in.html> (Erişim tarihi: 11.05.2018)

[15] <http://campus.murraystate.edu/academic/faculty/wlyle/325/Chapter28.pdf> (Erişim tarihi: 11.05.2018)

[16] <https://coderanch.com/t/456331/java/Creating-showing-JFrame-JFrame> (Erişim tarihi: 11.05.2018)

[17] <http://www.codejava.net/java-se/swing/preventing-jframe-window-from-closing> (Erişim tarihi: 11.05.2018)

[18] <https://www.macs.hw.ac.uk/cs/java-swing-guidebook/?name=JCheckBox&page=1> (Erişim tarihi: 11.05.2018)

[19] <https://stackoverflow.com/questions/17366553/select-records-with-date-in-the-last-24-hours> (Erişim tarihi: 11.05.2018)

[20] <https://www.smartdraw.com/entity-relationship-diagram/> (Erişim tarihi: 12.05.2018)

[21] <https://www.techonthenet.com/sql_server/foreign_keys/foreign_delete.php> (Erişim tarihi: 15.05.2018)

7 - Kazanımlar

Bu projede MySQL veritabanı yönetim sistemi ve oluşturduğumuz Java projesi arasında bağlantı kurup, veritabanında kayıt ekleme, silme, güncelleme ve filtreleme gibi temel işlemlerin GUI uygulaması şeklinde nasıl yapılacağını öğrendik. Projeyi geliştirirken birçok aşamada hatayla karşılaştık ve genel konseptlerden çok ayrıntılar üzerinde yoğunlaştık. Örneğin, hazırladığımız create-db-tables.sql dosyası ile veritabanını oluştururken, tabloları oluşturduğumuz kısımlarda depolama motoru (storage engine) olarak Navicat for MySQL programında editlediğimizde “InnoDB” yerine “MyISAM” default storage engine olarak kaldığı için Foreign Key özelliğinin olmadığını fark ettik ve bunu değiştirdik, foreign key tanımladıktan sonra Delete işlemini yapamadığımız için foreign key tanımlarken yazdığımız “ON DELETE NO ACTION” kısmını “ON DELETE SET NULL” olarak değiştirdik. Ayrıca MySQL 8.0.11 ile MySQL 5.7 arasındaki bazı farklılıklardan dolayı MySQL Workbench’te doğru yazdığımız bazı sorgular çalışmadığı için MySQL 5.7’ye geri döndük. Veritabanı kısmında buna benzer pek çok ayrıntıyla uğraşarak veritabanı yönetimi ile SQL konusundaki bilgi düzeyimizi arttırdık. Bunun dışında masaüstü uygulaması istendiği için Java Swing ve AWT API’leri üzerinde JFrame, JPanel, JLabel, JTextField, JButton, JTable, JComboBox, JCheckBox gibi özellikleri kullanıp birçok deneme yaparak bu konulardaki bilgi düzeyimizi de arttırdık.