İSTANBUL GELİŞİM ÜNİVERSİTESİ

Final Uygulama Projesi Raporu

Yıl / Dönem:

2023-2024 / Güz Yarıyılı

Ders Adı / Ders Kodu:

NESNE TABANLI PROGRAMLAMA II (BBP213)

Proje Değerlendirme Baremi / Genel Not Baremine Etkisi(%):

Final Uygulaması

Dersin Öğretim Elemanı:

Öğr. Gör. ÖZNUR AYAZOĞLU

Bölüm/Program:

İGMYO / BİLGİSAYAR PROGRAMCILIĞI (N.Ö.) – A Şubesi

Proje Grubu Adı / Konu:

MARİNA YÖNETİM SİSTEMİ

Proje Grubu Takım Üyeleri:

220111549-BURAK İŞLER

ÖDEV TANITIM FORMU

YAZAR ADI SOYADI : Burak İŞLER

ÖDEVİN DİLİ : Türkçe

ÖDEVİN ADI : Marina Yönetim Sistemi

BÖLÜM : Bilgisayar Teknolojileri

PROGRAM : Bilgisayar Programcılığı

ÖDEVİN TÜRÜ : Final

ÖDEVİN TES. TARİHİ : 29.12.2023

SAYFA SAYISI : 20

ÖDEV DANIŞMANI : Öğr. Gör. Öznur AYAZOĞLU

KABUL VE ONAY SAYFASI

220111549 numaralı Burak İŞLER'ın Marina Yönetim Sistemi adlı çalışması, benim tarafımdan Final ödevi olarak kabul edilmiştir.

Öğretim Görevlisi

Öznur AYAZOĞLU

ÖZET

Bu projenin amacı, çalışanların bir marina yat park sistemini etkileşimli bir arayüz üzerinden yönetmelerini sağlamaktadır. Görevliler, yat giriş, çıkış, güncelleme ve listeleme işlemlerini gerçekleştirebilirler.

İÇİNDEKİLER

İçindekiler

ÖNSÖZ

Dünyanın her yerinde bir çok büyük firma ve marina mevcut, yaptığım program firma ve marinaların yönetim sistemlerinde iş işleyişine oldukça etki edecektir. Firmadaki ve marinalardaki karışıklıklar sona erecektir. İstanbul Gelişim Üniversitesi'nde öğretim görevlisi olarak bizlerle bilgi ve birikimini paylaştığı için değerli hocam Öznur AYAZOĞLU'na teşekkürlerimi sunuyorum.

SİSTEM NE İŞE YARIYOR?

Bu sistem, bir marina işletmesini yönetmeye yardımcı olan bir programdır. Kullanıcılar, bu program aracılığıyla marina içindeki yatların giriş, çıkış, güncelleme ve listeleme işlemlerini yapabilirler. Yani, marina çalışanları, yatlarla ilgili bilgileri kaydedebilir, güncelleyebilir ve sorgulayabilirler. Program, Java programlama dili kullanılarak oluşturulmuş ve basit bir grafik arayüzü ile kullanıcılara sunulmuştur. Bu sayede, marina işlemleri daha düzenli ve anlaşılır bir şekilde gerçekleştirilebilir.

HEDEF KİTLE

Bu projenin hedef kitlesi, marina işletmeleri veya yat sahipleri gibi denizcilikle ilgili işlerle uğraşan kişilerdir. Bu tip kullanıcılar, marinalarda bulunan yatların giriş-çıkış takibini yapmak, bilgileri güncellemek veya genel olarak marina operasyonlarını yönetmek isteyebilirler. Proje, bu kullanıcı kitlesine, etkileşimli bir grafik arayüz aracılığıyla kolay ve kullanıcı dostu bir deneyim sunmayı amaçlar. Java programlama dilinde geliştirilen bu proje, marina işletmeleri veya yat sahipleri tarafından kullanılabilecek bir yazılım aracı olarak düşünülmüştür.

ZAMAN ÇİZELGESİ

1. Gün (20/12/2023): Proje tasarımı ve gereksinim analizi, veri tabanı tasarımı, Java projesinin temel yapısının oluşturulması.

2. Gün (21/12/2023): Grafik kullanıcı arayüzünün oluşturulması, yat girişi ve çıkışı için temel işlevselliğin eklenmesi.

3. Gün (22/12/2023): Yat güncelleme ve listeleme işlevselliğinin eklenmesi, veri tabanı entegrasyonunun tamamlanması.

4. Gün (23/12/2023): Projenin test aşaması, hataların düzeltilmesi ve kullanıcı dostu bir deneyim sağlanması.

5. Gün (24/12/2023): Dokümantasyon hazırlığına başlanması, projenin genel incelemesi ve düzenlemeler.

6. Gün (25/12/2023): Dokümantasyonun tamamlanması, son incelemelerin yapılması, proje tamamlanması ve sunuma hazırlık.

PROJE EKİBİ VE YÖNETİMİ

1. **Süleyman Demir:**
    - **Rol:** Proje Lideri ve Ana Geliştirici
    - **Görevler:**
        - Projenin genel yönetiminden sorumlu olacak.
        - Diğer ekip üyeleriyle koordinasyon sağlayacak.
        - Temel Java kodlama ve grafik ekran tasarımı üzerinde çalışacak.
        - Veri tabanı tasarımına katkıda bulunacak.
2. **Batuhan Sevin:**
    - **Rol:** Ana Geliştirici
    - **Görevler:**
        - Projenin genel yönetiminden sorumlu olacak.
        - Diğer ekip üyeleriyle koordinasyon sağlayacak.
        - Temel Java kodlama ve grafik ekran tasarımı üzerinde çalışacak.
        - Veri tabanı tasarımına katkıda bulunacak.
3. **Gizem Nur Çetin:**
    - **Rol:** Ana Geliştirici
    - **Görevler:**
        - Projenin genel yönetiminden sorumlu olacak.
        - Diğer ekip üyeleriyle koordinasyon sağlayacak.
        - Temel Java kodlama ve grafik ekran tasarımı üzerinde çalışacak.
        - Veri tabanı tasarımına katkıda bulunacak.
4. **Burak İşler:**
    - **Rol:** Ana Geliştirici
    - **Görevler:**
        - Projenin genel yönetiminden sorumlu olacak.
        - Diğer ekip üyeleriyle koordinasyon sağlayacak.
        - Temel Java kodlama ve grafik ekran tasarımı üzerinde çalışacak.
        - Veri tabanı tasarımına katkıda bulunacak.

KOD ADIMLARI

```
package main;

import javax.imageio.ImageIO;
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
import java.sql.*;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
```

Konumuz ilk başta bu aşamayla başlıyor. Bu aşamayı detaylandırmak gerekirse kütüphane import ettiğimiz kısım bu kısım kodun içinde ihtiyaç duyulan tüm kütüphaneler burada eklenmiş halde.

```
public class MarinaYatParkSistemi extends JFrame {

    private Connection baglanti;
    private JButton giristusu;
    private JButton cikistusu;
    private JButton listelemetusu;
    private JButton guncellemetusu;
    private JButton programikapat;

    public String veritabanikullaniciadi = "root";
    public String veritabanisifresi = "";

    public MarinaYatParkSistemi() {

        setTitle("Marina Yat Park Sistemi");

        // arkaplan resmini eklemek için aşağıdaki işlemleri yapıyoruz
        try {
            // img klasörü altındaki arkaplan2.png resmini okuyoruz
            BufferedImage img = ImageIO.read(new File( pathname: "img/arkaplan.jpg"));
            ImageIcon icon = new ImageIcon(img);
            JLabel arkaplan = new JLabel(icon);
            // arkaplan resmini panelin içine ekliyoruz
            setContentPane(arkaplan);
            // bu kod eklenmezse sadece arkaplan resmi görünür. paneldeki tüm itemların görünmesi iç
            // itemları yazım sırasına göre ekliyor. Böylece önce arkaplan resmi, sonra itemlar ekle
            setLayout(new FlowLayout());
        } catch (IOException e) {
            e.printStackTrace();
        }

        setSize( width: 1024, height: 580);
        // panelin boyutunu sabitledik
        setResizable(false);
        // ekranın ortasında açılması için aşağıdaki kodu yazdık
        setLocationRelativeTo(null);

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);


        JPanel panel = new JPanel();

        giristusu = new JButton( text: "Giriş Kaydı");
        giristusu.setBorder(BorderFactory.createLineBorder(Color.GREEN, thickness: 2));
```

```
giristusu = new JButton( text: "Giriş Kaydı");
giristusu.setBorder(BorderFactory.createLineBorder(Color.GREEN, thickness: 2));
giristusu.setPreferredSize(new Dimension( width: 150, height: 50));
// giris btn arkaplan eklmek için aşağıdaki kodu yazdık
ImageIcon icongiris = new ImageIcon( filename: "img/giris.png");
Image resimgiris = icongiris.getImage();
// resimi küçültmek için aşağıdaki kodu yazdık
Image boyutlandirilangirisresim = resimgiris.getScaledInstance( width: 60, height: 30, Image.SCALE_SMOOTH);
giristusu.setIcon(new ImageIcon(boyutlandirilangirisresim));

cikistusu = new JButton( text: "Çıkış Yap");
cikistusu.setBorder(BorderFactory.createLineBorder(Color.RED, thickness: 2));
cikistusu.setPreferredSize(new Dimension( width: 150, height: 50));
// cikis btn arkaplan eklmek için aşağıdaki kodu yazdık
ImageIcon iconcikis = new ImageIcon( filename: "img/cikis.png");
Image resimcikis = iconcikis.getImage();
Image boyutlandirilancikisresim = resimcikis.getScaledInstance( width: 60, height: 30, Image.SCALE_SMOOTH);
cikistusu.setIcon(new ImageIcon(boyutlandirilancikisresim));

listelemetusu = new JButton( text: "Yatları Listele");
listelemetusu.setBorder(BorderFactory.createLineBorder(Color.CYAN, thickness: 2));
listelemetusu.setPreferredSize(new Dimension( width: 150, height: 50));

guncellemetusu = new JButton( text: "Güncelle");
guncellemetusu.setBorder(BorderFactory.createLineBorder(Color.YELLOW, thickness: 2));
guncellemetusu.setPreferredSize(new Dimension( width: 150, height: 50));

programikapat = new JButton( text: "Uygulamayı Kapat");
programikapat.setBorder(BorderFactory.createLineBorder(Color.PINK, thickness: 2));
programikapat.setPreferredSize(new Dimension( width: 150, height: 50));

giristusu.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) { yatGirisYap(); }
});

cikistusu.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) { yatCikisYap(); }
});

listelemetusu.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) { yatlariListele(); }
```

```
guncellemetusu.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) { yatlariGuncelle(); }
});


programikapat.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) { programikapat(); }
});

panel.add(giristusu);
panel.add(cikistusu);
panel.add(listelemetusu);
panel.add(guncellemetusu);
panel.add(programikapat);

add(panel);

setVisible(false);

if (isValidLogin()) {
    setVisible(true);
} else {
    JOptionPane.showMessageDialog( parentComponent: this,  message: "Hatalı Giriş Sistem Kapatılıyor...");
    System.exit( status: 0);
}
```

"Giriş Kaydı" butonu yat girişi yapmak için kullanılır.

"Çıkış Yap" butonu yat çıkışı yapmak için kullanılır.

"Yatları Listele" butonu mevcut yatları listeler.

"Güncelle" butonu yat bilgilerini günceller.

"Uygulamayı Kapat" butonu uygulamayı kapatır.

Arkaplan Resmi: Uygulamanın arkaplanında bir resim kullanıldı.

Veritabanı Bağlantısı: Uygulama veritabanına bağlanıyor.

Giriş Kontrolü: isValidLogin fonksiyonu, girişin kontrolünü sağlar. Başarılı bir giriş yoksa hata mesajı gösterir ve uygulama kapatılır.

Buton İşlevleri: Her buton, ilgili işlevi gerçekleştiren bir metodla ilişkilidir.

"Giriş Kaydı" butonuna tıklanması durumunda yatGirisYap() metodunu çağırır.

"Çıkış" butonuna tıklanması durumunda yatCikisYap() metodunu çağırır.

"Güncelle" butonuna tıklanması durumunda yatlariGuncelle() metodunu çağırır.

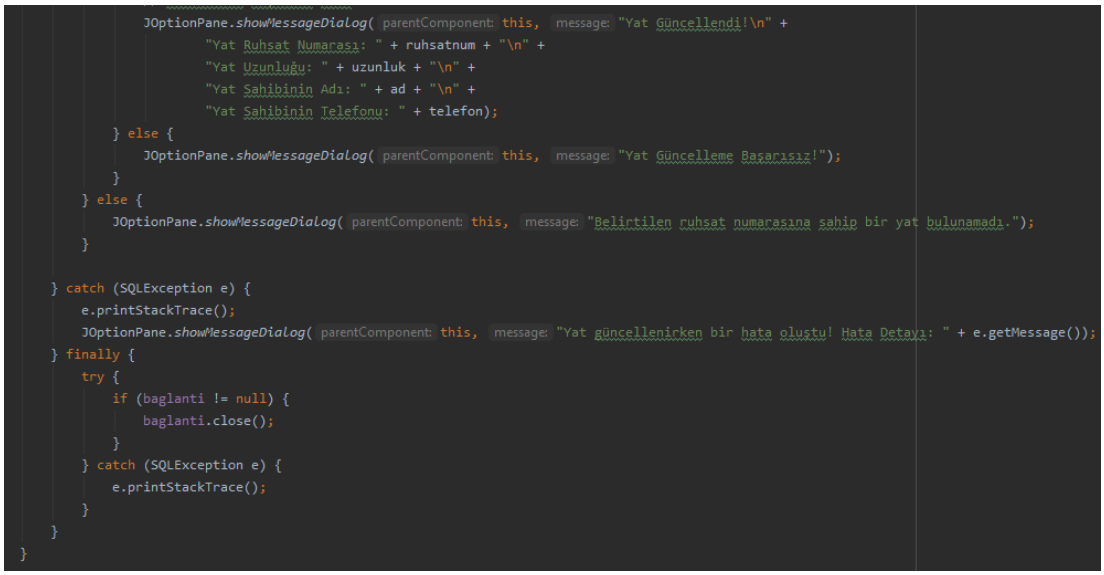
"Sistemi Kapat" butonuna tıklanması durumunda programiKapat() metodunu çağırır.

"Yatları Listele" butonuna tıklanması durumunda yatlarilistele() metodunu çağırır.

```java
private void programikapat() {

    JOptionPane.showMessageDialog( parentComponent: this,  message: "Sistem Kapatılıyor...");

    System.exit( status: 0);

}
```

Burada ise, programı kapatma metodunu ekledik ilk önce, ekrana sistemin kapatıldığına dair bilgi verdikten sonra komutla beraber sistemi kapatıyor.

```java
private void yatlariGuncelle() {

    String ruhsatnum = JOptionPane.showInputDialog( parentComponent: this,  message: "Güncellemek istediğiniz Yat Ruhsat Numarası:");

    if (ruhsatnum == null || ruhsatnum.isEmpty()) {
        return; // Kullanıcı işlemi iptal etti veya boş bir değer girdi
    }

    try {
        // Veritabanında ruhsatnum kontrolü
        baglanti = DriverManager.getConnection( url: "jdbc:mysql://localhost:3306/yatlar", veritabanikullaniciadi, veritabanisifresi);
        PreparedStatement checkStatement = baglanti.prepareStatement( sql: "SELECT * FROM yatlar WHERE ruhsatnum = ?");
        checkStatement.setString( parameterIndex: 1, ruhsatnum);
        ResultSet baglantisonuc = checkStatement.executeQuery();

        if (baglantisonuc.next()) {
            // Belirtilen ruhsatnum'a sahip bir yat bulundu yat için güncellenmiş bilgileri alıyorum
            String uzunluk = JOptionPane.showInputDialog( parentComponent: this,  message: "Yat Uzunluğu:");
            String ad = JOptionPane.showInputDialog( parentComponent: this,  message: "Yat Sahibinin Adı:");
            String telefon = JOptionPane.showInputDialog( parentComponent: this,  message: "Yat Sahibinin Telefonu:");

            if (uzunluk == null || ad == null || telefon == null) {
                return; // Kullanıcı işlemi iptal etti veya boş bir değer girdi
            }

            // Veritabanında güncelleme işlemini yap
            PreparedStatement updateStatement = baglanti.prepareStatement(
                    sql: "UPDATE yatlar SET uzunluk = ?, ad = ?, telefon = ? WHERE ruhsatnum = ?"
            );
            updateStatement.setString( parameterIndex: 1, uzunluk);
            updateStatement.setString( parameterIndex: 2, ad);
            updateStatement.setString( parameterIndex: 3, telefon);
            updateStatement.setString( parameterIndex: 4, ruhsatnum);

            int sguncelleme = updateStatement.executeUpdate();

            if (sguncelleme > 0) {
                // Güncelleme başarılı oldu
                JOptionPane.showMessageDialog( parentComponent: this,  message: "Yat Güncellendi!\n" +
                        "Yat Ruhsat Numarası: " + ruhsatnum + "\n" +
                        "Yat Uzunluğu: " + uzunluk + "\n" +
                        "Yat Sahibinin Adı: " + ad + "\n" +
                        "Yat Sahibinin Telefonu: " + telefon);
            } else {
                JOptionPane.showMessageDialog( parentComponent: this,  message: "Yat Güncelleme Başarısız!");
```



Bu metodda ise, kullanıcının veritabanındaki yatın bilgilerini güncellemesini sağlar. Kullanıcıdan güncellenmek istenen yatın ruhsat numarası (ruhsatnum) alınır. Fakat kullanıcı bu işlemi iptal eder ya da boş bir değer girerse işlem sonlandırılır. Veritabanında belirtilen ruhsat numarasına sahip bir yatın olup olmadığı kontrol edilir. Ama belirtilen ruhsat numarasına sahip bir yat bulunursa kullanıcıdan güncellenmiş bilgiler (uzunluk, ad, telefon) alınır. Kullanıcı işlemi iptal eder yada boş bir değer girerse işlem sonlandırılır. Veritabanında bu yatın bilgileri güncellenir. Güncelleme işlemi başarılıysa kullanıcıya güncellenen yatın bilgilerini içeren bir bilgi gösterilir. Güncelleme işlemi başarısızsa kullanıcıya bir hata mesajı gösterilir. Belirtilen ruhsat numarasına sahip bir yat yoksa kullanıcıya bilgi verilir. Bu metod, veritabanı bağlantısı sql sorguları ve joptionpane kullanıcı arayüzü kullanarak yat bilgilerini güncellemek için tasarlandı. Aynı zamandaysa olası hata durumlarını kontrol eder ve kullanıcıya uygun mesajları gösterir.

```
private void yatGirisYap() {

    String ruhsatnum = JOptionPane.showInputDialog( parentComponent: this,  message: "Yat Ruhsat numarası:");
    String uzunluk = JOptionPane.showInputDialog( parentComponent: this,  message: "Yat Uzunluğu:");
    String ad = JOptionPane.showInputDialog( parentComponent: this,  message: "Yat Sahibinin adı:");
    String telefon = JOptionPane.showInputDialog( parentComponent: this,  message: "Yat Sahibinin Telefonu:");

    if (ruhsatnum == null || uzunluk == null || ad == null || telefon == null) {
        return; // Kullanıcı pencereyi kapattı veya iptal etti
    }

    if (ruhsatnum.isEmpty() || uzunluk.isEmpty() || ad.isEmpty() || telefon.isEmpty()) {
        JOptionPane.showMessageDialog( parentComponent: this,  message: "Lütfen tüm alanları doldurun!");
    } else {
        try {
            // Veritabanında ruhsatnum ile kayıt var mı kontrolü yapıyorum
            baglanti = DriverManager.getConnection( url: "jdbc:mysql://localhost:3306/yatlar", veritabanikullaniciadi, veritabanisifresi);
            PreparedStatement kontroldata = baglanti.prepareStatement( sql: "SELECT * FROM yatlar WHERE ruhsatnum = ?");
            kontroldata.setString( parameterIndex: 1, ruhsatnum);
            ResultSet sonuckontrol = kontroldata.executeQuery();

            if (sonuckontrol.next()) {
                JOptionPane.showMessageDialog( parentComponent: this,  message: "Bu numaraya sahip yat zaten kayıtlı.");
            } else {
                // Yatın kaydedilmediği durumda işleme devam et
                PreparedStatement yatgirisdata = baglanti.prepareStatement( sql: "INSERT INTO yatlar VALUES (?, ?, ?, ?, ?)");
                yatgirisdata.setString( parameterIndex: 1, ruhsatnum);
                yatgirisdata.setString( parameterIndex: 2, uzunluk);
                yatgirisdata.setString( parameterIndex: 3, ad);
                yatgirisdata.setString( parameterIndex: 4, telefon);

                Date girisSaati = new Date();
                SimpleDateFormat tarihformati = new SimpleDateFormat( pattern: "yyyy-MM-dd HH:mm:ss");
                yatgirisdata.setString( parameterIndex: 5, tarihformati.format(girisSaati));

                yatgirisdata.executeUpdate();

                JOptionPane.showMessageDialog( parentComponent: this,  message: "Yat Giriş Yaptı!\nYat Ruhsat numarası: " + ruhsatnum +
                        "\nYat uzunlugu: " + uzunluk + "\nYat Sahibinin Adı: " + ad + "\nYat Sahibinin Telefonu: " + telefon +
                        "\nGiriş Saati: " + tarihformati.format(girisSaati));
            }
        } catch (SQLException e) {
            e.printStackTrace();
```

```
                yatgirisdata.executeUpdate();

                JOptionPane.showMessageDialog( parentComponent: this,  message: "Yat Giriş Yaptı!\nYat Ruhsat numarası: " + ruhsatnum +
                        "\nYat uzunlugu: " + uzunluk + "\nYat Sahibinin Adı: " + ad + "\nYat Sahibinin Telefonu: " + telefon +
                        "\nGiriş Saati: " + tarihformati.format(girisSaati));
            }
        } catch (SQLException e) {
            e.printStackTrace();
            JOptionPane.showMessageDialog( parentComponent: this,  message: "Yat eklenirken bir hata oluştu! Hata Detayı: " + e.getMessage());
        } finally {
            try {
                baglanti.close(); // Bağlantıyı kapat
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
}
```

Bu metod, kullanıcının yeni bir yat girişi yapmasını sağlar. Kullanıcıdan yeni yatın ruhsat numarası (ruhsatnum), uzunluğu (uzunluk), sahibinin adı (ad) ve telefon numarası (telefon) alınır. Kullanıcı penceresini kapattır yada işlemi iptal ederse metod sonlanır. Fakat tüm alanlar doldurulmadıysa ekrana bir uyarı mesajı gösterilir ve metod sonlanır. Veritabanında verilen ruhsat numarasına sahip bir yatın olup olmadığı kontrol edilir. Ama belirtilen ruhsat numarasına sahip bir yat zaten kayıtlıysa ekrana bir uyarı mesajı gösterilir. Eğer sözkonusu ruhsat numarasına sahip

bir yat kayıtlı değilse yeni yat kaydı veritabanına eklenir. Eklenen yatın bilgileri (ruhsat numarası, uzunluk, sahibinin adı, telefon numarası, giriş saati) formatıyla ekrana bir pencerede gösterilir. Veritabanına bağlanmada bir sorun olursa ekrana bir hata mesajı gösterilir.

```
private boolean isValidLogin() {

    JTextField kullaniciadialani = new JTextField();
    JPasswordField sifrealani = new JPasswordField();

    Object[] message = {
            "Kullanıcı Adı:", kullaniciadialani,
            "Şifre:", sifrealani
    };

    int girissecenek = JOptionPane.showConfirmDialog( parentComponent: this, message,  title: "Giriş", JOptionPane.OK_CANCEL_OPTION);

    if (girissecenek == JOptionPane.OK_OPTION) {
        String kullaniciadi = kullaniciadialani.getText();
        String sifre = new String(sifrealani.getPassword());

        try {
            // kullanıcı tablosunda girilen kullanıcı adı ve şifre ile kayıt var mı kontrolü yapıyorum
            baglanti = DriverManager.getConnection( url: "jdbc:mysql://localhost:3306/yatlar", veritabanikullaniciadi, veritabanisifresi);
            PreparedStatement preparedStatement = baglanti.prepareStatement( sql: "SELECT * FROM kullanicilar WHERE kullaniciadi = ? AND sifre = ?");
            preparedStatement.setString( parameterIndex: 1, kullaniciadi);
            preparedStatement.setString( parameterIndex: 2, sifre);

            ResultSet sonucset = preparedStatement.executeQuery();

            if (sonucset.next()) {
                // Giriş başarılı
                return true;
            } else {
                // Giriş başarısız
                return false;
            }

        } catch (SQLException e) {
            e.printStackTrace();
            JOptionPane.showMessageDialog( parentComponent: this,  message: "Veritabanı bağlantısı kurulamadı!");
            return false;
        }
    }
    return false;
}
```

Bu metoddaysa kullanıcının girişi için bir pencere oluşturur ve kullanıcıdan giriş bilgilerini alır. Alınan bu bilgileri sql veritabanındaki "kullanicilar" tablosu ile karşılaştırarak doğru olup olmadığını kontrol eder.

JTextField ve JPasswordField kullanarak bir giriş penceresi oluşturduk. Kullanıcıdan kullanıcı adı (kullaniciadialani) ve şifre (sifrealani)'yi alır. Kullanıcıya OK ve CANCEL düğmeleri içeren ekranı gösterir (JOptionPane.showConfirmDialog). Eğer kullanıcı OK düğmesine basarsa giriş bilgisini alır, CANCEL düğmesine tıklarsa veya pencereyi kapatırsa, false döner. Bilgileri alır sonra bu bilgileri veritabanındaki "kullanicilar" tablosundaki kayıtlarla kıyaslar. Veritabanında kullanıcı adı ve şifre ile bir kayıt bulunursa, giriş başarılı kabul edilir ve true döner. Eğer eşleşen bir kayıt yoksa giriş başarısız olur ve false döner. Veritabanına erişim veya sorgu yaparken bir hata olursa ekrana hata mesajı gösterir ve false döner.

```
private void yatCikisYap() {

    String ruhsatnum = JOptionPane.showInputDialog( parentComponent: this,  message: "Yat Ruhsat Numarası:");
    String yakit = JOptionPane.showInputDialog( parentComponent: this,  message: "Yakıt (Litre):");

    if (ruhsatnum == null || yakit == null) {
        // Kullanıcı pencereyi kapattı veya iptal etti
        return;
    }

    if (ruhsatnum.isEmpty() || yakit.isEmpty()) {
        JOptionPane.showMessageDialog( parentComponent: this,  message: "Lütfen ruhsat numarasını ve yakıt bilgilerini doldurun!");
    } else {
        try {
            // Veritabanında ruhsatnum kontrolü
            baglanti = DriverManager.getConnection( url: "jdbc:mysql://localhost:3306/yatlar", veritabanikullaniciadi, veritabanisifresi);
            PreparedStatement datakontrol = baglanti.prepareStatement( sql: "SELECT * FROM yatlar WHERE ruhsatnum = ?");
            datakontrol.setString( parameterIndex: 1, ruhsatnum);
            ResultSet sonucset = datakontrol.executeQuery();

            if (sonucset.next()) {
                // Veritabanında kayıt var. Yat classını veritabanından aldığım bilgileri constructor içinde kullanarak dolduruyorum.
                YatBilgisi yat = new YatBilgisi(
                        sonucset.getString( columnLabel: "uzunluk"),
                        sonucset.getString( columnLabel: "ad"),
                        sonucset.getString( columnLabel: "telefon"),
                        sonucset.getDate( columnLabel: "girisSaati")
                );

                Date cikisSaati = new Date();
                SimpleDateFormat tarihformatii = new SimpleDateFormat( pattern: "dd-MM-yyyy HH:mm:ss");
                String cikisSaatiStr = tarihformatii.format(cikisSaati);

                double ekUcret = 0;

                try {
                    double yakitMiktari = Double.parseDouble(yakit);
                    int uzunluk = Integer.parseInt(yat.getuzunluk());
                    double saatlikUcret;

                    if (uzunluk < 15) {
                        saatlikUcret = 7.0;
                    } else if (uzunluk > 15 && uzunluk <= 25) {
                        saatlikUcret = 12.5;
                    } else {
                        saatlikUcret = 16.0;
                    }

                    ekUcret = (cikisSaati.getTime() - yat.getGirisSaati().getTime()) / (1000 * 60 * 60.0) * saatlikUcret + yakitMiktari * 35.45;
                } catch (NumberFormatException ex) {
                    JOptionPane.showMessageDialog( parentComponent: this,  message: "Geçerli bir yakıt miktarı girin!");
                    return;
                }

                String bicimlendirilmisucret = String.format("%.2f", ekUcret);

                // Çıkış bilgilerini göster

                JOptionPane.showMessageDialog( parentComponent: this,  message: "Yat Çıkış Yaptı \n" +
                        "Yat ruhsat numarası: " + ruhsatnum + "\n" +
                        "Yat Uzunluğu: " + yat.getuzunluk() + "\n" +
                        "Yat Sahibinin Adı: " + yat.getAd() + "\n" +
                        "Telefon Numarası: " + yat.getTelefon() + "\n" +
                        "Giriş Tarihi: " + yat.getGirisSaati() + "\n" +
                        "Çıkış Saati: " + cikisSaatiStr + "\n" +
                        "Yakıt Miktarı: " + yakit + " litre" + "\n" +
                        "Ücret: " + bicimlendirilmisucret + " TL");

                // Veritabanından kaydı sil
                PreparedStatement kaydisil = baglanti.prepareStatement( sql: "DELETE FROM yatlar WHERE ruhsatnum = ?");
                kaydisil.setString( parameterIndex: 1, ruhsatnum);
                kaydisil.executeUpdate();

            } else {
                JOptionPane.showMessageDialog( parentComponent: this,  message: "Veritabanında bu numaraya sahip bir kayıt bulunamadı.");
            }
        } catch (SQLException e) {
            e.printStackTrace();
            JOptionPane.showMessageDialog( parentComponent: this,  message: "Çıkış yapılırken bir hata oluştu! Hata Detayı: " + e.getMessage());
        } finally {
            try {
                baglanti.close(); // Bağlantıyı kapat
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
```

Bu metodda bir yatın çıkış işlemini gerçekleştiriyoruz. Kullanıcıdan yatın ruhsat numarası ve çıkışta alınmışsa eğer yakıt miktarını alır. Sonra bu bilgileri kullanarak veritabanındaki kaydı getirir ve yatın giriş bilgileri ile çıkış bilgilerini kullanarak hesaplamalar yapar. Kullanıcıdan yat ruhsat numarasını (ruhsatnum) ve kullanılan yakıt miktarını (yakit) JOptionPane.showInputDialog'la alıyoruz. Ama kullanıcı pencereyi kapatırsa yada işlemi iptal ederse metodun geri kalanı çalıştırılmadan durur. Girilen bilgiler boş mu diye kontrol edilir boşsa eğer kullanıcıya alanları doldurmasını için mesaj göönderiyoruz. Veritabanında yatın ruhsat numarasını kontrol ediyoruz (SELECT * FROM yatlar WHERE ruhsatnum = ?).Ve böyle bir kayıt varsa bilgileri bir YatBilgisi'ne doldurur. Yatın giriş saati ve çıkış saati arasındaki süreyi hesaplayıp bu süre içindeki ücreti ve yakıt masrafını hesaplar. Hesaplanan ücret ve diğer bilgilerle bir (JOptionPane.showMessageDialog) ile bu bilgileri ekrana gösteriyoruz. Veritabanından ilgili yat kaydını siliyoruz çünkü kayıt artık veritabanımızda bulunmuyor (DELETE FROM yatlar WHERE ruhsatnum = ?).

```
private void yatlariListele() {

    try {
        // Veritabanına bağlan
        baglanti = DriverManager.getConnection( url: "jdbc:mysql://localhost:3306/yatlar", veritabanikullaniciadi, veritabanisifresi);
        PreparedStatement listelemeyatlari = baglanti.prepareStatement( sql: "SELECT * FROM yatlar");
        ResultSet listelemeset = listelemeyatlari.executeQuery();

        // Veritabanındaki kayıtları içeren bir liste oluştur
        List<String[]> yatListesi = new ArrayList<>();
        while (listelemeset.next()) {
            String ruhsatnum = listelemeset.getString( columnLabel: "ruhsatnum");
            String uzunluk = listelemeset.getString( columnLabel: "uzunluk");
            String ad = listelemeset.getString( columnLabel: "ad");
            String telefon = listelemeset.getString( columnLabel: "telefon");
            String girisSaatiStr = listelemeset.getString( columnLabel: "girisSaati");

            yatListesi.add(new String[]{ruhsatnum, uzunluk, ad, telefon, girisSaatiStr});
        }

        if (yatListesi.isEmpty()) {
            JOptionPane.showMessageDialog( parentComponent: this,  message: "Marinada kayıtlı yat bulunmamaktadır.");
            return;
        }

        // Liste verilerini kullanarak Array oluştur. toArray() metoduna parametre olarak boş bir String dizisi gönderiyoruz.
        // Bu sayede toArray() metodunun dönüş tipi String[][] oluyor. yazım kalıbı olarak toArray(new String[0][]) kullanılabilir.
        String[][] data = yatListesi.toArray(new String[0][]);
        String[] kolonadi = {"Yat Ruhsat Numarası", "Yat uzunluğu", "Yat Sahibinin Adı", "Telefon Numarası", "Giriş Saati"};

        JTable jtablo = new JTable(data, kolonadi);
        jtablo.getTableHeader().setBackground(Color.GRAY);

        JScrollPane kaydirma = new JScrollPane(jtablo);

        JOptionPane pane = new JOptionPane(kaydirma, JOptionPane.PLAIN_MESSAGE, JOptionPane.DEFAULT_OPTION,  icon: null, new Object[]{},  initialValue: null);
        JDialog diyalog = pane.createDialog( parentComponent: this,  title: "Marinadaki Yatlar");
        diyalog.setSize( width: 900,  height: 300);
        diyalog.setVisible(true);

    } catch (SQLException e) {
        e.printStackTrace();
        JOptionPane.showMessageDialog( parentComponent: this,  message: "Yatları listelerken bir hata oluştu! Hata Detayı: " + e.getMessage());
    } finally {
        try {
```

```
        } catch (SQLException e) {
            e.printStackTrace();
            JOptionPane.showMessageDialog( parentComponent: this,  message: "Yatları listelerken bir hata oluştu! Hata Detayı: " + e.getMessage());
        } finally {
            try {
                if (baglanti != null) {
                    baglanti.close(); // Bağlantıyı kapat
                }
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
```

Bu metodda veritabanındaki "yatlar" tablosundaki kayıtları ekrana getiriyoruz. Jdbc kullanarak veritabanına bağlantı kuruyoruz Bilgileri veritabanikullaniciadi ve veritabanisifresi değişkenlerinden alıyoruz. SELECT * FROM yatlar sorgusunu kullanıp veritabanındaki "yatlar" tablosundaki tüm kayıtları getiriyoruz. Sonuçları ResultSet üzerinden alıyoruz. Veritabanındaki kayıtları içeren bir liste oluşturuyoruz (List<String[]> yatListesi). Her yatın bilgilerini bu listede bir dizi olarak tutuyoruz. Eğer yatListesi boşsa yani marina kayıtlı yat yoksa ekrana "Marinada kayıtlı yat bulunmamaktadır." mesajını gösteriyoruz ve metodu sonlandırıyoruz. Eğer yatListesi boş değilse bu bilgileri kullanıp bir tablo oluşturuyoruz. JTable kullanarak tabloyu oluşturuyoruz ve tabloya başlık ekliyoruz. Tabloyu JScrollPane içine yerleştirdik ki eğer tablo çok büyükse kullanıcılar kaydırarak görüntüleyebilir. JOptionPane ve JDialog kullanıp bir pencere oluşturduk. O pencerede oluşturduğumuz tabloyu içeren JScrollPane'i gösteriyoruz. Eğer veritabanı işlemleri yapılırken bir hata olursa ekrana "Yatları listelerken bir hata oluştu!" mesajını göstertiyoruz. Finally bloğunda ise bağlantıyı kapatıyoruz. Bu da her durumda bağlantının sorunsuz kapatılmasını sağlar.

```
    public static void main(String[] args) {

        SwingUtilities.invokeLater(new Runnable() {
            @Override
            public void run() {

                new MarinaYatParkSistemi();
            }
        });
    }
}
```

Bu main metodundaysa Java programının başlatıyoruz ve arayüzünün oluşturuyoruz. SwingUtilities.invokeLater(new Runnable() {...}) kısmında Swingi içeren bir

grafiksel arayüzü uygulamasını başlatıyoruz Grafiksel arayüz işlemleri genellikle Event Dispatch Thread üzerinden gerçekleştiriyoruz. Burda ise arayüz oluşturma işlemlerini EDT içinde gerçekleştirmek için kullanıyoruz. new Runnable() {...} kodunda, Runnable arayüzünü yapan anonim bir sınıf oluşturuyoruz. Bu sınıfın içine de run metodu tanımlıyoruz. run metodu arayüz oluşturma işlemini içeriyor. new MarinaYatParkSistemi() koduyla MarinaYatParkSistemi sınıfından bir nesne oluşturuyoruz. Sonuç olarak main metodunun içindeki kodlarla Swing uygulamasını başlatıyoruz.

```java
package main;

import java.text.SimpleDateFormat;
import java.util.Date;

class YatBilgisi {

    private String uzunluk;
    private String ad;
    private String telefon;
    private Date girisSaati;


    public YatBilgisi(String uzunluk, String ad, String telefon, Date girisSaati) {
        this.uzunluk = uzunluk;
        this.ad = ad;
        this.telefon = telefon;
        this.girisSaati = girisSaati;
    }

    public String getuzunluk() {
        return uzunluk;
    }

    public String getAd() {
        return ad;
    }

    public String getTelefon() {
        return telefon;
    }

    public Date getGirisSaati() {
        return girisSaati;
}

    public String getGirisSaatiStr() {

        SimpleDateFormat tarihformati2 = new SimpleDateFormat( pattern: "dd-MM-yyyy HH:mm:ss");

        return tarihformati2.format(girisSaati);
    }
}
```

Bu classta yatlarla ilgili bilgileri tutuyor ve bu bilgilere erişmek için kullanıyoruz.

Özellikler:

uzunluk: Yatın uzunluğunu temsil eden String.

ad: Yat sahibinin adını temsil eden String.

telefon: Yat sahibinin telefon numarasını temsil eden String.

girisSaati: Yatın giriş saati bilgisini temsil eden Date.

Constructor:

public YatBilgisi(String uzunluk, String ad, String telefon, Date girisSaati): Bu metot ile YatBilgisinden bir nesne oluşturuyoruz ve yatın temel bilgilerini alıyoruz.

Getter Metotları:

getuzunluk(): Yatın uzunluğunu döndüren String getter metodu.

getAd(): Yat sahibinin adını döndüren String getter metodu.

getTelefon(): Yat sahibinin telefon numarasını döndüren String getter metodu.

getGirisSaati(): Yatın giriş saati bilgisini döndüren Date getter metodu.

Özel Getter Metot:

getGirisSaatiStr(): Yatın giriş saatini tarih formatına çeviren bir String getter metodu. SimpleDateFormat kullanarak tarihi "dd-MM-yyyy HH:mm:ss" formatına dönüştürmemize yarıyor.

```
CREATE DATABASE IF NOT EXISTS `yatlar` DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;
USE `yatlar`;
SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
START TRANSACTION;
SET time_zone = "+00:00";
```

Burada ise Veritabanının sql kodlarını ekliyoruz ‘yatlar’ adında bir database oluşturuyoruz.

```sql
CREATE TABLE `kullanicilar` (
  `kullaniciadi` varchar(50) NOT NULL,
  `sifre` varchar(50) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Burada ise ‘Kulllanicilar’ adında bir tablo oluşturup içine ‘kllaniciadi’ ve ‘sifre’ kolonu oluşturuyoruz.

```sql
INSERT INTO `kullanicilar` (`kullaniciadi`, `sifre`) VALUES
('admin', 'admin123');
```

Burada ise ‘kulanicilar’ tablosundaki ‘kullaniciadi’ ve ‘sifre’ kolonlarına değer atıyoruz.

```sql
CREATE TABLE `yatlar` (
  `ruhsatnum` varchar(20) NOT NULL,
  `uzunluk` varchar(10) NOT NULL,
  `ad` varchar(150) NOT NULL,
  `telefon` varchar(30) NOT NULL,
  `girisSaati` datetime NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--
-- Tablo döküm verisi `yatlar`
--

INSERT INTO `yatlar` (`ruhsatnum`, `uzunluk`, `ad`, `telefon`, `girisSaati`) VALUES
('01FFT234', '19.', 'Aleyna Çelik', '+905436780912', '2023-12-09'),
('34ZAP944', '24.', 'Ahmet Akarca', '+905455673210', '2023-03-05'),
('35JKL012', '36.', 'Süleyman Demir', '+905444146953', '2023-04-08'),
('545', '5454', 'dsfsdfs', '54541', '2023-12-26'),
('GH789', '14.', 'Bill Guorgini', '+01674587628', '2023-04-07'),
('MNO345', '42', 'John Affrold', '+464718293', '2023-09-03'),
('PR678', '8', 'Ludgozky Malentin', '+901234567895', '2023-10-04'),
('ST901', '14', 'Ludmilla Kratiskov', '+7288790543', '2023-12-24'),
('ZR3788', '9', 'Adina Lorens', '+451234567891', '2023-02-01'),
('ZR4725', '24', 'Igor', '+365848754', '2023-12-26');

--
-- Dökümü yapılmış tablolar için indeksler
--


--
-- Tablo için indeksler `kullanicilar`
--
ALTER TABLE `kullanicilar`
  ADD PRIMARY KEY (`kullaniciadi`);

--
-- Tablo için indeksler `yatlar`
--
ALTER TABLE `yatlar`
  ADD PRIMARY KEY (`ruhsatnum`);
COMMIT;
```

Burada ise ‘yatlar’ tablosunu oluşturup içine ‘ruhsatnum’,’uzunluk’,’ad’,’telefon’,’girisSaati’ kolonları ekliyoruz ve yatlar tablosuna veri girişi yapıyoruz.

KAYNAKÇA

- https://www.yusufsezer.com.tr/java-awt-swing-ve-javafx/
- https://www.tasarimkodlama.com/java-programlama/java-jtable-kullanimi-ve-ornekleri/
- https://javasalih.blogspot.com/2016/11/java-systemexit-program-istenilen-yerde.html
- https://belenyasin.com/2017/07/03/java-tarih-formatlama/
- https://harun.xyz/java/java-ders-6-cikti-bicimleme-type-casting/
- https://blog.burakkutbay.com/otopark-bilgi-sistemi-projesi.html/
- https://github.com/YusufDagdeviren/JavaIleVeritabani
- https://www.furkanozbay.com/2014/05/pencereyejframe-resim-ekleme.html