

MAT335E

Project: Tavuk

Depo Takip Sistemi





**Muhammed Selman
Demir**



Emirhan akır



İçindekiler

- Proje Amacı
- Sistem Mimarisi
- Kullanıcı Roller ve İşlevler
- Arayüz Tasarımı
- Kod Örnekleri
- Sonuç ve Kullanıcı Hedef Kitleler

Projenin Amacı



Projenin Amacı

- Depodaki malların takibini sağlamak
- Kullanıcıların ürünleri transfer edebilmeleri
- Yöneticilerin kullanıcı yönetimi yapabilmeleri

Sistem Mimarisi



Sistem Mimarisı

- Java, MYSQL ve Swing kullanılarak geliştirilmiştir.
- Kullanıcı ve ürün bilgileri veritabanında saklanır.
- Arayüz Swing kütüphanesi kullanılarak oluşturulmuştur.

Kullanıcı Roller ve İşlevler





Yönetici

- Kullanıcı ekleme/silme
- Depoya ürün ekleme/silme
- Kullanıcıya ürün transfer etme



Normal Kullanıcı

- Depolar arası transfer
- Ürün düzenleme

Arayüz Tasarımı



Kullanıcı Yönetimi (Administrator)

- Kullanıcı ekleme-çıkarma işlemleri burada düzenlenir.
- Sadece administrator kullanıcı ekleyip çıkarabilir.

ID	Kullanıcı Adı	Şifre	Rol
1	admin	admin123	yonetici
2	user1	user123	kullanici
3	emirhan	tavuk	kullanici
4	selman	123	yonetici



The image shows a screenshot of a login window titled "Giriş Ekranı". The window has a standard Windows-style title bar with a minimize button, a maximize button, and a close button. The main area of the window is divided into three sections. The first section is labeled "Kullanıcı Adı:" and contains a text input field. The second section is labeled "Şifre:" and contains a text input field. The third section is a button labeled "Giriş".

Kullanıcı Adı:	<input type="text"/>
Şifre:	<input type="password"/>
<input type="button" value="Giriş"/>	

Giriş Ekranı

- Sisteme administrator tarafından eklenmeniz gerekmektedir.
- Çok basit bir şekilde kullanıcı adı ve şifre girerek sisteme giriş yapabilirsiniz.

Ürün Yönetimi

- Adım 1: Kullanılacak depo seçilir.
- Adım 2: Ürünün adı ve miktarı girilir.
- Adım 3:
 - Ekle: Ürünü depoya ekler.
 - Sil: O depodaki bir ürünü kaldırır.
 - Transfer Et: Seçilen depodaki bir miktar ürünü, başka bir depoya aktarır.
 - Sat: Alıcıya belli bir miktar satış gerçekleştirir.

Ürün Yönetimi

Depo Seçimi:

Depo 1

Ürün Adı:

Miktar:

Ekle

Sil

Transfer Et

Sat

ID	Ürün Adı	Miktar	Depo ID
23	tavuk	31	1
24	tavuk	35	2
25	Jack Daniels 70 cl	3	1

Ürün Transferi

Kaynak Depo:	Depo 2
Hedef Depo:	Depo 1
Ürün Adı:	Bremen
Miktar:	10
Transfer Et	

Ürün Transferi

- Kaynak depo ve hedef depo seçilir.
- Ürün adı ve miktarı girilir.
- Transfer Et butonuna tıklanınca işlem gerçekleşir.

Kod Örnekleri



```
package Project;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class DatabaseConnection {
    private static final String URL = "jdbc:mysql://localhost:3306/depo_takip";
    private static final String USER = "root";
    private static final String PASSWORD = "emirhan";

    public static Connection getConnection() {
        Connection connection = null;
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            connection = DriverManager.getConnection(URL, USER, PASSWORD);
        } catch (ClassNotFoundException | SQLException e) {
            e.printStackTrace();
        }
        return connection;
    }
}
```

DataBaseConnection Class



```
package Project;

public class Depo {
    private int depoId;
    private String depoAdi;

    // Constructor with all fields
    public Depo(int depoId, String depoAdi) {
        this.depoId = depoId;
        this.depoAdi = depoAdi;
    }

    // Constructor without depoId (for new entries)
    public Depo(String depoAdi) {
        this.depoAdi = depoAdi;
    }

    // Getter and Setter methods
    public int getDepoId() {
        return depoId;
    }

    public void setDepoId(int depoId) {
        this.depoId = depoId;
    }

    public String getDepoAdi() {
        return depoAdi;
    }

    public void setDepoAdi(String depoAdi) {
        this.depoAdi = depoAdi;
    }

    // Override toString method for JComboBox display
    @Override
    public String toString() {
        return depoAdi;
    }
}
```

Depo Class



```
package Project;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

public class DepoDAO {
    public void addDepo(Depo depo) {
        String sql = "INSERT INTO depolar (depo_adi) VALUES (?)";
        try (Connection connection = DatabaseConnection.getConnection();
            PreparedStatement statement = connection.prepareStatement(sql)) {
            statement.setString(1, depo.getDepoAdi());
            statement.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    public void deleteDepo(int id) {
        String sql = "DELETE FROM depolar WHERE depo_id = ?";
        try (Connection connection = DatabaseConnection.getConnection();
            PreparedStatement statement = connection.prepareStatement(sql)) {
            statement.setInt(1, id);
            statement.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

AddDepo ve DeleteDepo Methodları

DepoDAO Class



DepoList Methodu

```
public List<Depo> getAllDepolar() {  
    List<Depo> depolar = new ArrayList<>();  
    String sql = "SELECT * FROM depolar";  
    try (Connection connection = DatabaseConnection.getConnection();  
        PreparedStatement statement = connection.prepareStatement(sql);  
        ResultSet resultSet = statement.executeQuery()) {  
        while (resultSet.next()) {  
            int depoId = resultSet.getInt("depo_id");  
            String depoAdi = resultSet.getString("depo_adi");  
            depolar.add(new Depo(depoId, depoAdi));  
        }  
    } catch (SQLException e) {  
        e.printStackTrace();  
    }  
    return depolar;  
}
```

DepoDao Class




```
package Project;

public class Kullanici {
    private int id;
    private String kullaniciAdi;
    private String sifre;
    private String rol;

    public Kullanici(int id, String kullaniciAdi, String sifre, String rol) {
        this.id = id;
        this.kullaniciAdi = kullaniciAdi;
        this.sifre = sifre;
        this.rol = rol;
    }

    public Kullanici(String kullaniciAdi, String sifre, String rol) {
        this.kullaniciAdi = kullaniciAdi;
        this.sifre = sifre;
        this.rol = rol;
    }

    public int getId() {
        return id;
    }

    public String getKullaniciAdi() {
        return kullaniciAdi;
    }

    public String getSifre() {
        return sifre;
    }

    public String getRol() {
        return rol;
    }
}
```

Kullanıcı Class



```
package Project;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

public class KullaniciDAO {
    // Kullanıcı ekleme metodu
    public void addKullanici(Kullanici kullanıcı) {
        String sql = "INSERT INTO kullanicilar (kullanici_adi, sifre, rol) VALUES (?, ?, ?)";
        try (Connection connection = DatabaseConnection.getConnection();
            PreparedStatement statement = connection.prepareStatement(sql)) {
            statement.setString(1, kullanıcı.getKullaniciAdi());
            statement.setString(2, kullanıcı.getSifre());
            statement.setString(3, kullanıcı.getRol());
            statement.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    // Kullanıcı silme metodu
    public void deleteKullanici(int id) {
        String sql = "DELETE FROM kullanicilar WHERE id = ?";
        try (Connection connection = DatabaseConnection.getConnection();
            PreparedStatement statement = connection.prepareStatement(sql)) {
            statement.setInt(1, id);
            statement.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

AddKullanici ve DeleteKullanici Methodları

KullaniciDAO Class



ListKullanici ve GetKullanici Methodları

```
// Tüm kullanıcıları listeleme metodu
public List<Kullanici> getAllKullanicilar() {
    List<Kullanici> kullanicilar = new ArrayList<>();
    String sql = "SELECT * FROM kullanicilar";
    try (Connection connection = DatabaseConnection.getConnection();
        PreparedStatement statement = connection.prepareStatement(sql);
        ResultSet resultSet = statement.executeQuery()) {
        while (resultSet.next()) {
            int id = resultSet.getInt("id");
            String kullanıcıAdi = resultSet.getString("kullanici_adi");
            String sifre = resultSet.getString("sifre");
            String rol = resultSet.getString("rol");
            kullanicilar.add(new Kullanici(id, kullanıcıAdi, sifre, rol));
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return kullanicilar;
}

// Kullanıcıyı kullanıcı adı ve şifre ile doğrulama metodu
public Kullanici getKullaniciByKullaniciAdiVeSifre(String kullanıcıAdi, String sifre) {
    String sql = "SELECT * FROM kullanicilar WHERE kullanici_adi = ? AND sifre = ?";
    try (Connection connection = DatabaseConnection.getConnection();
        PreparedStatement statement = connection.prepareStatement(sql)) {
        statement.setString(1, kullanıcıAdi);
        statement.setString(2, sifre);
        try (ResultSet resultSet = statement.executeQuery()) {
            if (resultSet.next()) {
                int id = resultSet.getInt("id");
                String rol = resultSet.getString("rol");
                return new Kullanici(id, kullanıcıAdi, sifre, rol);
            }
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return null;
}
```

KullaniciDAO Class



```
package Project;

public class Urun {
    private int urunId;
    private String urunAdi;
    private int miktar;
    private int depoId;

    // Constructor with all fields
    public Urun(int urunId, String urunAdi, int miktar, int depoId) {
        this.urunId = urunId;
        this.urunAdi = urunAdi;
        this.miktar = miktar;
        this.depoId = depoId;
    }

    // Constructor without urunId (for new entries)
    public Urun(String urunAdi, int miktar, int depoId) {
        this.urunAdi = urunAdi;
        this.miktar = miktar;
        this.depoId = depoId;
    }

    // Getter and Setter methods
    public int getUrunId() {
        return urunId;
    }

    public void setUrunId(int urunId) {
        this.urunId = urunId;
    }

    public String getUrunAdi() {
        return urunAdi;
    }

    public void setUrunAdi(String urunAdi) {
        this.urunAdi = urunAdi;
    }

    public int getMiktar() {
        return miktar;
    }

    public void setMiktar(int miktar) {
        this.miktar = miktar;
    }

    public int getDepoId() {
        return depoId;
    }

    public void setDepoId(int depoId) {
        this.depoId = depoId;
    }
}
```

Urun Class




```

public class UrunDAO {
    private String username;

    public UrunDAO(String username) {
        this.username = username;
    }

    // Ürün ekleme veya güncelleme metodu
    public void addOrUpdateUrun(Urun urun) {
        String checkSql = "SELECT * FROM urunler WHERE urun_adi = ? AND depo_id = ?";
        String updateSql = "UPDATE urunler SET miktar = miktar + ? WHERE urun_adi = ? AND depo_id = ?";
        String insertSql = "INSERT INTO urunler (urun_adi, miktar, depo_id) VALUES (?, ?, ?)";

        try (Connection connection = DatabaseConnection.getConnection();
            PreparedStatement checkStatement = connection.prepareStatement(checkSql)) {

            // Önce ürünün depoda olup olmadığını kontrol edelim
            checkStatement.setString(1, urun.getUrunAdi());
            checkStatement.setInt(2, urun.getDepoId());
            ResultSet resultSet = checkStatement.executeQuery();

            if (resultSet.next()) {
                // Ürün zaten depoda varsa, miktarını güncelleyelim
                try (PreparedStatement updateStatement = connection.prepareStatement(updateSql)) {
                    updateStatement.setInt(1, urun.getMiktar());
                    updateStatement.setString(2, urun.getUrunAdi());
                    updateStatement.setInt(3, urun.getDepoId());
                    updateStatement.executeUpdate();
                    LogHelper.logAction("Ürün güncellendi: " + urun.getUrunAdi() + ", miktar: " + urun.getMiktar() + ", depo: " + urun.getDepoId(), username);
                }
            } else {
                // Ürün depoda yoksa, yeni bir ürün ekleyelim
                try (PreparedStatement insertStatement = connection.prepareStatement(insertSql)) {
                    insertStatement.setString(1, urun.getUrunAdi());
                    insertStatement.setInt(2, urun.getMiktar());
                    insertStatement.setInt(3, urun.getDepoId());
                    insertStatement.executeUpdate();
                    LogHelper.logAction("Ürün eklendi: " + urun.getUrunAdi() + ", miktar: " + urun.getMiktar() + ", depo: " + urun.getDepoId(), username);
                }
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}

```

UrunDAO Class

AddDepo Methodu



```
// Ürün silme metodu
public void deleteUrun(int id, String urunAdi, int depoId) {
    String sql = "DELETE FROM urunler WHERE urun_id = ?";
    try (Connection connection = DatabaseConnection.getConnection();
        PreparedStatement statement = connection.prepareStatement(sql)) {
        statement.setInt(1, id);
        statement.executeUpdate();
        LogHelper.logAction("Ürün silindi: " + urunAdi + ", depo: " + depoId, username);
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

UrunDAO Class

DeleteUrun Metodu




```
// Ürün satma metodu
public void sellUrun(String urunAdi, int miktar, int depoId) {
    String checkSql = "SELECT miktar FROM urunler WHERE urun_adi = ? AND depo_id = ?";
    String updateSql = "UPDATE urunler SET miktar = miktar - ? WHERE urun_adi = ? AND depo_id = ?";
    String deleteSql = "DELETE FROM urunler WHERE urun_adi = ? AND depo_id = ?";

    try (Connection connection = DatabaseConnection.getConnection();
        PreparedStatement checkStatement = connection.prepareStatement(checkSql)) {

        checkStatement.setString(1, urunAdi);
        checkStatement.setInt(2, depoId);
        ResultSet resultSet = checkStatement.executeQuery();

        if (resultSet.next()) {
            int mevcutMiktar = resultSet.getInt("miktar");
            if (mevcutMiktar >= miktar) {
                // Ürünün miktarını güncelle
                try (PreparedStatement updateStatement = connection.prepareStatement(updateSql)) {
                    updateStatement.setInt(1, miktar);
                    updateStatement.setString(2, urunAdi);
                    updateStatement.setInt(3, depoId);
                    updateStatement.executeUpdate();
                    LogHelper.logAction("Ürün satıldı: " + urunAdi + ", miktar: " + miktar + ", depo: " + depoId, username);
                }

                // Eğer miktar sıfır veya negatif olduysa ürünü sil
                if (mevcutMiktar - miktar <= 0) {
                    try (PreparedStatement deleteStatement = connection.prepareStatement(deleteSql)) {
                        deleteStatement.setString(1, urunAdi);
                        deleteStatement.setInt(2, depoId);
                        deleteStatement.executeUpdate();
                        LogHelper.logAction("Ürün tamamen satıldı ve silindi: " + urunAdi + ", depo: " + depoId, username);
                    }
                } else {
                    throw new SQLException("Yeterli miktarda ürün yok.");
                }
            } else {
                throw new SQLException("Depoda bu ürün bulunamadı.");
            }
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

UrunDAO Class

SellUrun Metodu

```
// Tüm ürünleri listeleme metodu
public List<Urun> getAllUrunler() {
    List<Urun> urunler = new ArrayList<>();
    String sql = "SELECT * FROM urunler";
    try (Connection connection = DatabaseConnection.getConnection();
        PreparedStatement statement = connection.prepareStatement(sql);
        ResultSet resultSet = statement.executeQuery()) {
        while (resultSet.next()) {
            int urunId = resultSet.getInt("urun_id");
            String urunAdi = resultSet.getString("urun_adi");
            int miktar = resultSet.getInt("miktar");
            int depoId = resultSet.getInt("depo_id");
            urunler.add(new Urun(urunId, urunAdi, miktar, depoId));
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return urunler;
}
```

UrunDAO Class

ListUrun Metodu



```

// Ürün transfer metodu
public void transferUrun(String urunAdi, int miktar, int kaynakDepoId, int hedefDepoId) {
    // Kaynak depodaki ürünü kontrol et
    String checkSql = "SELECT miktar FROM urunler WHERE urun_adi = ? AND depo_id = ?";
    String updateKaynakSql = "UPDATE urunler SET miktar = miktar - ? WHERE urun_adi = ? AND depo_id = ?";
    String updateHedefSql = "UPDATE urunler SET miktar = miktar + ? WHERE urun_adi = ? AND depo_id = ?";
    String insertHedefSql = "INSERT INTO urunler (urun_adi, miktar, depo_id) VALUES (?, ?, ?)";

    try (Connection connection = DatabaseConnection.getConnection();
        PreparedStatement checkStatement = connection.prepareStatement(checkSql)) {
        checkStatement.setString(1, urunAdi);
        checkStatement.setInt(2, kaynakDepoId);
        ResultSet resultSet = checkStatement.executeQuery();

        if (resultSet.next()) {
            int mevcutMiktar = resultSet.getInt("miktar");
            if (mevcutMiktar >= miktar) {
                // Kaynak depodan miktarı düş
                try (PreparedStatement updateKaynakStatement = connection.prepareStatement(updateKaynakSql)) {
                    updateKaynakStatement.setInt(1, miktar);
                    updateKaynakStatement.setString(2, urunAdi);
                    updateKaynakStatement.setInt(3, kaynakDepoId);
                    updateKaynakStatement.executeUpdate();
                    LogHelper.logAction("Ürün transfer edildi: " + urunAdi + ", miktar: " + miktar + " kaynaktan: " + kaynakDepoId + " hedefe: " + hedefDepoId, username);
                }

                // Hedef depoda ürün var mı kontrol et
                try (PreparedStatement checkHedefStatement = connection.prepareStatement(checkSql)) {
                    checkHedefStatement.setString(1, urunAdi);
                    checkHedefStatement.setInt(2, hedefDepoId);
                    resultSet = checkHedefStatement.executeQuery();

                    if (resultSet.next()) {
                        // Hedef depoda ürün varsa miktarı güncelle
                        try (PreparedStatement updateHedefStatement = connection.prepareStatement(updateHedefSql)) {
                            updateHedefStatement.setInt(1, miktar);
                            updateHedefStatement.setString(2, urunAdi);
                            updateHedefStatement.setInt(3, hedefDepoId);
                            updateHedefStatement.executeUpdate();
                        }
                    } else {
                        // Hedef depoda ürün yoksa yeni kayıt ekle
                        try (PreparedStatement insertHedefStatement = connection.prepareStatement(insertHedefSql)) {
                            insertHedefStatement.setString(1, urunAdi);
                            insertHedefStatement.setInt(2, miktar);
                            insertHedefStatement.setInt(3, hedefDepoId);
                            insertHedefStatement.executeUpdate();
                        }
                    }
                }
            } else {
                throw new SQLException("Kaynak depoda yeterli miktarda ürün yok.");
            }
        } else {
            throw new SQLException("Kaynak depoda ürün bulunamadı.");
        }
    }
}

```

UrunDAO Class

TransferUrun Metodu

Arayüz Tasarım Kodları




```
package Project;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.List;

public class KullaniciYonetimi extends JFrame {
    private JTextField kullaniciAdiField;
    private JTextField sifreField;
    private JComboBox<String> rolComboBox;
    private JButton ekleButton;
    private JButton silButton;
    private JTable kullaniciTable;
    private KullaniciTableModel kullaniciTableModel;
    private KullaniciDAO kullaniciDAO;

    public KullaniciYonetimi() {
        kullaniciDAO = new KullaniciDAO();

        setTitle("Kullanıcı Yönetimi");
        setSize(600, 400);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new BorderLayout());

        JPanel panel = new JPanel(new GridLayout(4, 2));
        panel.add(new JLabel("Kullanıcı Adı:"));
        kullaniciAdiField = new JTextField();
        panel.add(kullaniciAdiField);

        panel.add(new JLabel("Şifre:"));
        sifreField = new JTextField();
        panel.add(sifreField);

        panel.add(new JLabel("Rol:"));
        rolComboBox = new JComboBox<>(new String[]{"yonetici", "kullanici"});
        panel.add(rolComboBox);

        ekleButton = new JButton("Ekle");
        ekleButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                String kullaniciAdi = kullaniciAdiField.getText();
                String sifre = sifreField.getText();
                String rol = (String) rolComboBox.getSelectedItem();
                Kullanici kullanici = new Kullanici(kullaniciAdi, sifre, rol);
                kullaniciDAO.addKullanici(kullanici);
                refreshTable();
            }
        });
        panel.add(ekleButton);
    }
}
```

Kullanıcı Yönetimi



```

silButton = new JButton("Sil");
silButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        int selectedRow = kullaniciTable.getSelectedRow();
        if (selectedRow != -1) {
            int kullaniciId = (int) kullaniciTableModel.getValueAt(selectedRow, 0);
            kullaniciDAO.deleteKullanici(kullaniciId);
            refreshTable();
        }
    }
});
panel.add(silButton);

add(panel, BorderLayout.NORTH);

kullaniciTableModel = new KullaniciTableModel();
kullaniciTable = new JTable(kullaniciTableModel);
add(new JScrollPane(kullaniciTable), BorderLayout.CENTER);

refreshTable();
}

private void refreshTable() {
    List<Kullanici> kullanicilar = kullaniciDAO.getAllKullanicilar();
    kullaniciTableModel.setKullanicilar(kullanicilar);
}

public static void main(String[] args) {
    SwingUtilities.invokeLater(new Runnable() {
        @Override
        public void run() {
            new KullaniciYonetimi().setVisible(true);
        }
    });
}
}

```

Kullanıcı Yönetimi Devamı


```
public GirişEkranı() {
    kullanıcıDAO = new KullanıcıDAO();

    setTitle("Giriş Ekranı");
    setSize(300, 150);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setLayout(new GridLayout(3, 2));

    add(new JLabel("Kullanıcı Adı:"));
    kullanıcıAdiField = new JTextField();
    add(kullanıcıAdiField);

    add(new JLabel("Şifre:"));
    şifreField = new JPasswordField();
    add(şifreField);

    girişButton = new JButton("Giriş");
    girişButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            String kullanıcıAdi = kullanıcıAdiField.getText();
            String şifre = new String(şifreField.getPassword());
            Kullanıcı kullanıcı = kullanıcıDAO.getKullanıcıByKullanıcıAdiVeŞifre(kullanıcıAdi, şifre);
            if (kullanıcı != null) {
                if ("yonetici".equals(kullanıcı.getRol())) {
                    KullanıcıYonetimi kullanıcıYonetimi = new KullanıcıYonetimi();
                    kullanıcıYonetimi.setLocationRelativeTo(null); // Pencereyi ekranın ortasında açar
                    kullanıcıYonetimi.setVisible(true);
                } else {
                    UrunYonetimi urunYonetimi = new UrunYonetimi(kullanıcıAdi);
                    urunYonetimi.setLocationRelativeTo(null); // Pencereyi ekranın ortasında açar
                    urunYonetimi.setVisible(true);
                }
            }
            dispose();
        } else {
            JOptionPane.showMessageDialog(GirişEkranı.this, "Geçersiz kullanıcı adı veya şifre", "Hata", JOptionPane.ERROR_MESSAGE);
        }
    });
    add(girişButton);
}
```

Giriş Ekranı



```
public static void main(String[] args) {  
    SwingUtilities.invokeLater(new Runnable() {  
        @Override  
        public void run() {  
            GirişEkranı girişEkranı = new GirişEkranı();  
            girişEkranı.setLocationRelativeTo(null); // Pencereyi ekranın ortasında açar  
            girişEkranı.setVisible(true);  
        }  
    });  
}
```

Giriş Ekranı Devamı



```
package Project;

import javax.swing.table.AbstractTableModel;
import java.util.List;

public class UrunTableModel extends AbstractTableModel {
    private final String[] columnNames = {"ID", "Ürün Adı", "Miktar", "Depo ID"};
    private List<Urun> urunler;

    @Override
    public int getRowCount() {
        return urunler == null ? 0 : urunler.size();
    }

    @Override
    public int getColumnCount() {
        return columnNames.length;
    }

    @Override
    public Object getValueAt(int rowIndex, int columnIndex) {
        Urun urun = urunler.get(rowIndex);
        switch (columnIndex) {
            case 0:
                return urun.getUrunId();
            case 1:
                return urun.getUrunAdi();
            case 2:
                return urun.getMiktar();
            case 3:
                return urun.getDepoId();
            default:
                return null;
        }
    }

    @Override
    public String getColumnName(int column) {
        return columnNames[column];
    }

    public void setUrunler(List<Urun> urunler) {
        this.urunler = urunler;
        fireTableDataChanged();
    }
}
```

Ürün Tablosu Modeli



```

package Project;

import javax.swing.table.AbstractTableModel;
import java.util.List;

public class KullaniciTableModel extends AbstractTableModel {
    private final String[] columnNames = {"ID", "Kullanıcı Adı", "Şifre", "Rol"};
    private List<Kullanici> kullanicilar;

    @Override
    public int getRowCount() {
        return kullanicilar == null ? 0 : kullanicilar.size();
    }

    @Override
    public int getColumnCount() {
        return columnNames.length;
    }

    @Override
    public Object getValueAt(int rowIndex, int columnIndex) {
        Kullanici kullanici = kullanicilar.get(rowIndex);
        switch (columnIndex) {
            case 0:
                return kullanici.getId();
            case 1:
                return kullanici.getKullaniciAdi();
            case 2:
                return kullanici.getSifre();
            case 3:
                return kullanici.getRol();
            default:
                return null;
        }
    }

    @Override
    public String getColumnName(int column) {
        return columnNames[column];
    }

    public void setKullanicilar(List<Kullanici> kullanicilar) {
        this.kullanicilar = kullanicilar;
        fireTableDataChanged();
    }
}

```

Kullanıcı Tablosu Modeli



```
package Project;

import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;

public class LogHelper {
    private static final String LOG_FILE = "log.txt";

    public static void logAction(String action, String username) {
        try (BufferedWriter writer = new BufferedWriter(new FileWriter(LOG_FILE, true))) {
            String timestamp = LocalDateTime.now().format(DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss"));
            writer.write(timestamp + " - " + username + " - " + action);
            writer.newLine();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Log Helper



Ürün Yönetimi

```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.List;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.List;

public class UrunYonetimi extends JFrame {
    private JComboBox<Depo> depoComboBox;
    private JTextField urunAdiField;
    private JTextField miktarField;
    private JButton ekleButton;
    private JButton silButton;
    private JButton transferButton;
    private JButton satButton;
    private JTable urunTable;
    private UrunTableModel urunTableModel;
    private UrunDAO urunDAO;
    private DepoDAO depoDAO;
    private String username;

    public UrunYonetimi(String username) {
        this.username = username;
        urunDAO = new UrunDAO(username);
        depoDAO = new DepoDAO();

        setTitle("Ürün Yönetimi");
        setSize(600, 400);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new BorderLayout());

        JPanel panel = new JPanel(new GridLayout(6, 2));
        panel.add(new JLabel("Depo Seçimi:"));
        depoComboBox = new JComboBox<>();
        List<Depo> depolar = depoDAO.getAllDepolar();
        for (Depo depo : depolar) {
            depoComboBox.addItem(depo);
        }
        panel.add(depoComboBox);

        panel.add(new JLabel("Ürün Adı:"));
        urunAdiField = new JTextField();
        panel.add(urunAdiField);

        panel.add(new JLabel("Miktar:"));
        miktarField = new JTextField();
        panel.add(miktarField);

        ekleButton = new JButton("Ekle");
        ekleButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                String urunAdi = urunAdiField.getText();
                int miktar = Integer.parseInt(miktarField.getText());
                Depo selectedDepo = (Depo) depoComboBox.getSelectedItem();
                Urun urun = new Urun(urunAdi, miktar, selectedDepo.getDepoId());
                urunDAO.addOrUpdateUrun(urun);
                refreshTable();
            }
        });
        panel.add(ekleButton);
    }
}
```


Ürün Yönetimi Devamı

```
silButton = new JButton("Sil");
silButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        int selectedRow = urunTable.getSelectedRow();
        if (selectedRow != -1) {
            int urunId = (int) urunTableModel.getValueAt(selectedRow, 0);
            String urunAdi = (String) urunTableModel.getValueAt(selectedRow, 1);
            int depoId = (int) urunTableModel.getValueAt(selectedRow, 2);
            urunDAO.deleteUrun(urunId, urunAdi, depoId);
            refreshTable();
        }
    }
});
panel.add(silButton);

transferButton = new JButton("Transfer Et");
transferButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        UrunTransferi urunTransferi = new UrunTransferi(UrunYonetimi.this, username); // Ürün transferi ekranına ana ekranı ve kullanıcı a
        urunTransferi.setLocationRelativeTo(null); // Pencereyi ekranın ortasında açar
        urunTransferi.setVisible(true);
    }
});
panel.add(transferButton);

satButton = new JButton("Sat");
satButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        String urunAdi = urunAdiField.getText();
        int miktar = Integer.parseInt(miktarField.getText());
        Depo selectedDepo = (Depo) depoComboBox.getSelectedItem();
        urunDAO.sellUrun(urunAdi, miktar, selectedDepo.getDepoId());
        refreshTable();
    }
});
panel.add(satButton);

add(panel, BorderLayout.NORTH);

urunTableModel = new UrunTableModel();
urunTable = new JTable(urunTableModel);
add(new JScrollPane(urunTable), BorderLayout.CENTER);

refreshTable();
}

public void refreshTable() {
    List<Urun> urunler = urunDAO.getAllUrunler();
    urunTableModel.setUrunler(urunler);
}

public static void main(String[] args) {
    SwingUtilities.invokeLater(new Runnable() {
        @Override
        public void run() {
            String username = "testUser"; // Test kullanıcı adı, burada giriş ekranından geçiş yapılabilir
            UrunYonetimi urunYonetimi = new UrunYonetimi(username);
            urunYonetimi.setLocationRelativeTo(null); // Pencereyi ekranın ortasında açar
            urunYonetimi.setVisible(true);
        }
    });
}
```

Ürün Transferi

```
package Project;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.List;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.List;

public class UrunTransferi extends JFrame {
    private JComboBox<Depo> kaynakDepoComboBox;
    private JComboBox<Depo> hedefDepoComboBox;
    private JTextField urunAdiField;
    private JTextField miktarField;
    private JButton transferButton;
    private UrunDAO urunDAO;
    private DepoDAO depoDAO;
    private UrunYonetimi urunYonetimi;
    private String username;

    public UrunTransferi(UrunYonetimi urunYonetimi, String username) {
        this.urunYonetimi = urunYonetimi;
        this.username = username;
        urunDAO = new UrunDAO(username);
        depoDAO = new DepoDAO();

        setTitle("Ürün Transferi");
        setSize(400, 300);
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE); // JFrame.EXIT_ON_CLOSE yerine DISPOSE_ON_CLOSE kullanıldı
        setLayout(new GridLayout(5, 2));

        add(new JLabel("Kaynak Depo:"));
        kaynakDepoComboBox = new JComboBox<>();
        List<Depo> depolar = depoDAO.getAllDepolar();
        for (Depo depo : depolar) {
            kaynakDepoComboBox.addItem(depo);
        }
        add(kaynakDepoComboBox);

        add(new JLabel("Hedef Depo:"));
        hedefDepoComboBox = new JComboBox<>();
        for (Depo depo : depolar) {
            hedefDepoComboBox.addItem(depo);
        }
        add(hedefDepoComboBox);

        add(new JLabel("Ürün Adı:"));
        urunAdiField = new JTextField();
        add(urunAdiField);

        add(new JLabel("Miktar:"));
        miktarField = new JTextField();
        add(miktarField);
    }
}
```

Ürün Transferi Devamı

```
transferButton = new JButton("Transfer Et");
transferButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        String urunAdi = urunAdiField.getText();
        int miktar = Integer.parseInt(miktarField.getText());
        Depo kaynakDepo = (Depo) kaynakDepoComboBox.getSelectedItem();
        Depo hedefDepo = (Depo) hedefDepoComboBox.getSelectedItem();

        if (kaynakDepo.getDepoId() == hedefDepo.getDepoId()) {
            JOptionPane.showMessageDialog(UrunTransferi.this, "Kaynak ve hedef depo aynı olamaz.", "Hata", JOptionPane.ERROR_MESSAGE);
        } else {
            urunDAO.transferUrun(urunAdi, miktar, kaynakDepo.getDepoId(), hedefDepo.getDepoId());
            JOptionPane.showMessageDialog(UrunTransferi.this, "Ürün başarıyla transfer edildi.", "Bilgi", JOptionPane.INFORMATION_MESSAGE);
            urunYonetimi.refreshTable(); // Ürün yönetimi tablosunu güncelle
        }
    }
});
add(transferButton);
}

public static void main(String[] args) {
    SwingUtilities.invokeLater(new Runnable() {
        @Override
        public void run() {
            UrunYonetimi urunYonetimi = new UrunYonetimi("testUser"); // Test kullanıcı adı, burada giriş ekranından geçiş yapılabilir
            UrunTransferi urunTransferi = new UrunTransferi(urunYonetimi, "testUser"); // Ürün transferi ekranı
            urunTransferi.setLocationRelativeTo(null); // Pencereyi ekranın ortasında açar
            urunTransferi.setVisible(true);
        }
    });
}
```

Sonuç ve Karşılaştırma



Problem

Koordinasyon



Stok



İş Sürecinde Karmaşa



Çözüm

Uygulamamız, transfer etme özelliğiyle çok daha kolay bir şekilde verilerin tutulmasını sağlamaktadır.

Uygulamamız, ekleme özelliğiyle stok durumlarını güncel tutulabilmektedir.

Uygulamamız; isrediğiniz kadar kullanıcı ve depo ekleyerek, temiz bir arayüzle bunları çok daha kolay hale getirmekte.

Projemiz, depolardaki malların etkili bir şekilde yönetimini sağlayarak işletmelerin operasyonel verimliliğini artırmaktadır.

Dinlediğiniz için Teşekkürler!

Emirhan Çakır
cakirem21@itu.edu.tr

Muhammed Selman Demir
demirmuha21@itu.edu.tr

