1. ANALİZ RAPORU (ANALYSIS REPORT)

DEPO YÖNETİM SİSTEMİ

Ders: Yazılım İnşaası – Iteration 1/2 Analiz Dokümanı Hazırlayanlar:

1. **Murat Efe ÖZOĞUL - 2410238037**
2. **Ömer ÇELİK - 2410238034**
3. **Berat GÜNGÖR - 2410238050**
4. **Teslim Tarihi: 04.01.2026**

İÇİNDEKİLER

---

1. GİRİŞ VE PROJENİN AMACI

Bu doküman, Karabük Üniversitesi Mühendislik Fakültesi Yazılım Mühendisliği bölümü öğrencileri tarafından geliştirilen Depo Yönetim Sistemi'nin analiz ve modelleme aşamasını kapsamaktadır.

Projenin temel amacı; manuel stok takibinin yarattığı zorlukları aşmak, depo yönetim süreçlerini dijitalleştirmek, ürün giriş-çıkışlarını, stok takibini ve raf düzenini hatasız ve hızlı bir şekilde gerçekleştirmektir. Sistem, veritabanı kurulumu gerektirmeyen dosya sistemi (Java Serialization) ile verilerin kalıcılığını sağlamaktadır.

---

2. SİSTEM GEREKSİNİMLERİ

2.1 Fonksiyonel Gereksinimler (Functional Requirements)

- **FR1 – Güvenli Yetkili Girişi: Sistem sadece yetkili "Mudur" girişi ile açılmalıdır. Hatalı şifre denemeleri sayılmalı, 3 kez hatalı giriş yapıldığında sistem güvenlik önlemi olarak kendini kapatmalıdır.**
- **FR2 – Raf Yönetimi ve Veri Tutarlılığı: Yönetici yeni raf ekleyebilmeli ve rafları silebilir. Ancak, bir raf silinmek istendiğinde sistem rafın dolu olup olmadığını kontrol etmeli, doluysa silme işlemini engellemelidir.**
- **FR3 – Akıllı Stok Dağıtımı: Depoya büyük miktarda ürün eklendiğinde tek bir rafın kapasitesi yetmezse, sistem "Parçalı Ekleme" önerisinde bulunmalı ve ürünleri boş kapasitesi olan diğer raflara otomatik dağıtmalıdır.**
- **FR4 – Raporlama: Depodaki ürünler listelenebilmeli, arama yapılabilmeli ve toplam doluluk oranı görüntülenebilmelidir.**

2.2 Fonksiyonel Olmayan Gereksinimler (Non-Functional Requirements)

- **NFR1 – Teknoloji: Sistem Java (JDK 17+) kullanılarak geliştirilmelidir.**
- **NFR2 – Mimari: Proje, Sorumlulukların Ayrılması (Separation of Concerns) ilkesine ve Katmanlı Mimariye uygun olmalıdır.**
- **NFR3 – Veri Saklama: Veriler `.bin` uzantılı dosyalarda (Binary Serialization) tutulmalıdır.**

2.3 Sistem Kısıtları (Constraints)

- **SQL Veritabanı kullanılmayacaktır.**
- **Uygulama masaüstü tabanlıdır (Console/GUI).**

3. AKTÖRLER

Aktör	Rol	Açıklama
Depo Müdürü (Mudur)	Baş Aktör	Sisteme giriş yapar, yetkili tek kişidir. Rafları yönetir, ürün giriş-çıkışını sağlar ve raporları inceler[10].

4. SİSTEM MODELLERİ

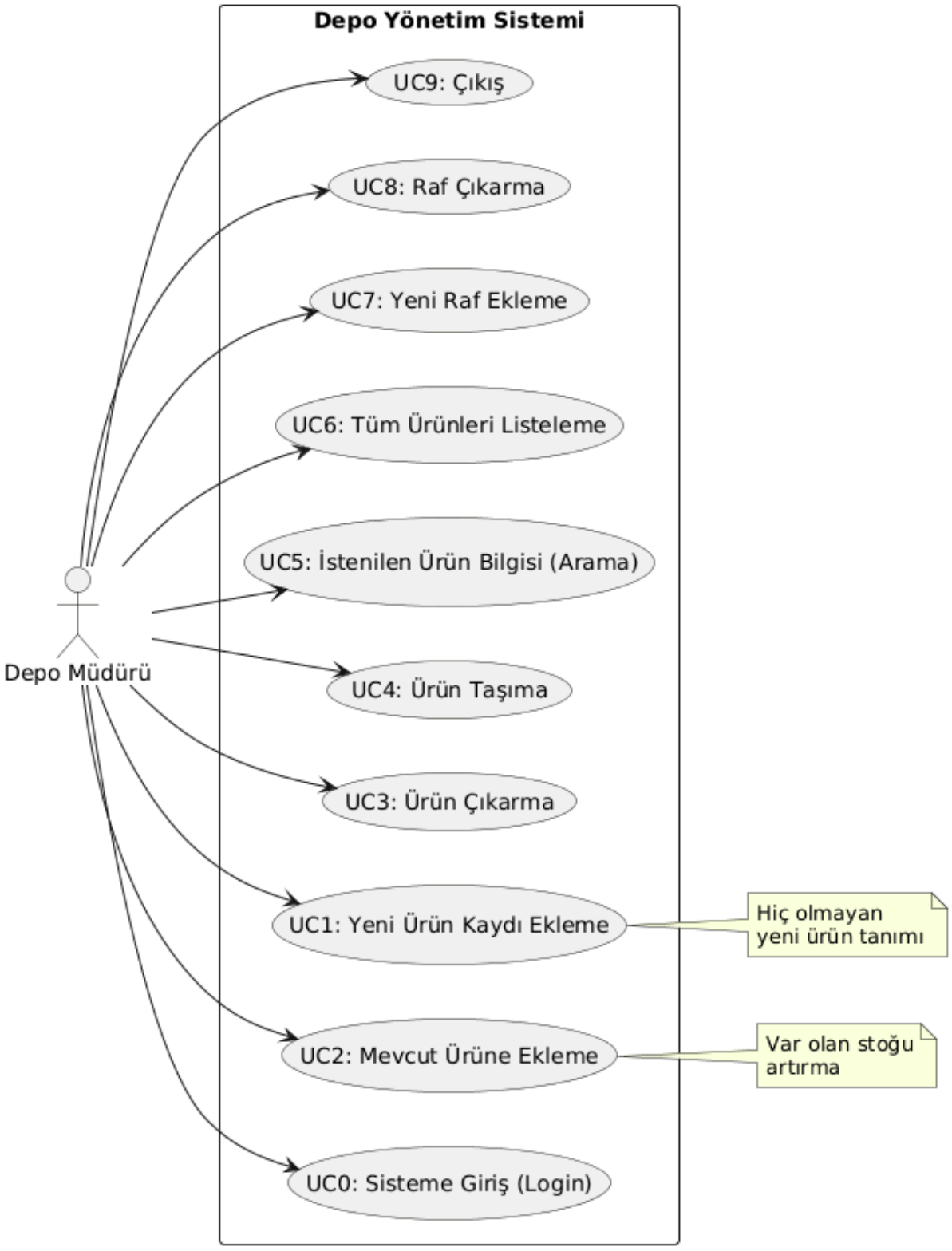
4.1 KULLANIM DURUMU MODELİ (Use Case Model)

Use Case'ler:

- **UC0: Sisteme Giriş (Login)**
- **UC1: Yeni Ürün Kaydı Ekleme**
- **UC2: Mevcut Ürüne Ekleme**
- **UC3: Ürün Çıkarma**
- **UC4: Ürün Taşıma**
- **UC5: İstenilen Ürün Bilgisi (Arama)**
- **UC6: Tüm Ürünleri Listeleme**
- **UC7: Yeni Raf Ekleme**
- **UC8: Raf Çıkarma**
- **UC9: Çıkış**

4.1.1 Kullanım Durumu Diyagramı

Aşağıdaki PlantUML kodu, sistemin fonksiyonel kapsamını görselleştirir.


Depo Yönetim Sistemi
UC9: Çıkış
UC8: Raf Çıkarma
UC7: Yeni Raf Ekleme
UC6: Tüm Ürünleri Listeleme
UC5: İstenilen Ürün Bilgisi (Arama)
UC4: Ürün Taşıma
UC3: Ürün Çıkarma
UC1: Yeni Ürün Kaydı Ekleme
Hiç olmayan
yeni ürün tanımı
UC2: Mevcut Ürüne Ekleme
Var olan stoğu
artırma
UC0: Sisteme Giriş (Login)
Depo Müdürü

4.1.2 Use Case Senaryo Açıklamaları

Aşağıdaki senaryolar, projenin kaynak kodlarında yer alan iş mantığına göre analiz edilmiştir.

UC0: SİSTEME GİRİŞ (Login ve Kayıt)

- **Kod Kaynağı: `MudurManager.java`**
- **Açıklama: Sistemin açılış aşamasıdır. Sistem ilk kez çalıştırılıyorsa kayıt, daha önce çalıştırılmışsa giriş ekranı gelir.**
- **Akış:**
    1. **`Main` sınıfı `mudurManager.sistemGiris()` metodunu çağırır.**
    2. **Kayıt Senaryosu: Eğer `mudur.bin` dosyası yoksa veya boşsa, `mudurKayit()` metodu devreye girer. Kullanıcıdan bir kullanıcı adı ve 4 basamaklı bir şifre (1000-9999 arası) istenir. Ardından depo kurulumu (`rafManager.ilkKurulum()`) yaptırılır.**
    3. **Giriş Senaryosu: Eğer yönetici kayıtlıysa `mudurGiris()` çalışır. Kullanıcıya 3 giriş hakkı tanınır. Şifre 3 kez yanlış girilirse "SİSTEM GÜVENLİK SEBEBİ İLE KAPATILIYOR" mesajı verilir ve program sonlanır.**

UC1: YENİ ÜRÜN KAYDI EKLEME

- **Kod Kaynağı: `UrunManager.java` (Metot: `urunKayitEkle`)**
- **Açıklama: Depo envanterinde bulunmayan, tamamen yeni bir ürünün sisteme tanıtılmasıdır.**
- **Akış:**
    1. **Kullanıcı Ürün Adı ve Seri Numarasını girer. Sistem, seri numarasının benzersiz olup olmadığını kontrol eder (`seriNoVarMi`).**
    2. **Eklenecek miktar ve hedef raf istenir.**
    3. **Kapasite Kontrolü: Eğer seçilen rafın kapasitesi yetersizse ancak depoda toplam boş yer varsa, sistem "Parçalı Ekleme" önerir.**
    4. **İnteraktif Dağıtım: Kullanıcı "Evet" derse, kalan miktarı hangi raflara ne kadar koyacağını kendisi seçer (`while` döngüsü ile).**
    5. **İşlem sonunda ürün, bulunduğu tüm rafların listesiyle (`rafListesi`) birlikte kaydedilir.**

UC2: MEVCUT ÜRÜNE EKLEME (Stok Güncelleme)

- **Kod Kaynağı: `UrunManager.java` (Metot: `urunEkleme`)**
- **Açıklama: Zaten kayıtlı olan bir ürünün stok miktarını artırma işlemidir.**
- **Akış:**
    1. **Kullanıcı Seri Numarası girer. Sistem ürünü bulamazsa hata verir.**
    2. **Eklenecek miktar ve hedef raf istenir.**
    3. **Akıllı Dağıtım: UC1'deki gibi, eğer tek bir rafın kapasitesi yetmezse, kullanıcıya interaktif olarak diğer raflara dağıtım yapma imkanı sunulur.**
    4. **Stok artışı ve raf kapasite düşüşü anlık olarak dosyalara işlenir.**

UC3: ÜRÜN ÇIKARMA

- **Kod Kaynağı:** `UrunManager.java` **(Metot:** `urunCikarma`**)**
- **Açıklama: Depodan ürün çıkışı yapılması veya satış/hurda işlemi.**
- **Akış:**
    1. **Seri numarası ile ürün bulunur.**
    2. **Çıkarılacak miktar girilir. Miktar negatif olamaz.**
    3. **Stok Kontrolü: Depodaki miktardan fazla ürün çıkarılamaz.**
    4. **İşlem sonrası rafın kapasitesi artırılır.**
    5. **Otomatik Silme: Eğer ürünün stoğu 0'a düşerse, ürün kaydı sistemden tamamen silinir. Stok 10'un altına düşerse kullanıcı uyarılır.**

UC4: ÜRÜN TAŞIMA

- **Kod Kaynağı:** `UrunManager.java` **(Metot:** `urunTasima`**)**
- **Açıklama: Bir ürünü bulunduğu raftan alıp başka bir rafa transfer etme işlemidir.**
- **Akış:**
    1. **Ürün Seri Numarası ile seçilir.**
    2. **Hedef raf kodu istenir.**
    3. **Kapasite Validasyonu: Hedef rafın boş kapasitesi, taşınacak ürün miktarı için yeterli değilse işlem engellenir ve kullanıcıdan başka bir raf seçmesi istenir.**
    4. **Başarılı taşımada; eski rafın kapasitesi artar, yeni rafın kapasitesi azalır ve ürünün raf bilgisi güncellenir.**

UC5: İSTENİLEN ÜRÜN BİLGİSİ (Arama)

- **Kod Kaynağı:** `Listeleme.java` **(Metot:** `listedeArama`**)**
- **Açıklama: Belirli bir ürünün detaylarını (Ad, Miktar, Bulunduğu Raflar) görüntüleme.**
- **Akış:**
    1. **Kullanıcıdan aranan ürünün seri numarası istenir.**
    2. **Liste taranır, eşleşme varsa detaylar yazdırılır.**
    3. **Ürün bulunamazsa kullanıcıya tekrar deneme hakkı verilir (5 hak). Haklar tükenirse ana menüye dönülür.**

UC6: TÜM ÜRÜNLERİ LİSTELEME

- **Kod Kaynağı:** `Listeleme.java` **(Metot:** `tumListeYazdir`**)**
- **Açıklama: Depodaki tüm envanterin dökümünün alınmasıdır.**
- **Akış:**
    1. **Kayıtlı tüm ürünler döngü ile gezilir.**
    2. **Her ürün için; Ad, Seri No, Miktar ve Bulunduğu Raflar (Örn: "1, 3") formatında ekrana çıktı verilir.**
    3. **Depo boşsa "HATA! Depoda herhangi bir ürün bulunmamaktadır" uyarısı verilir.**

UC7: YENİ RAF EKLEME

- **Kod Kaynağı: `RafManager.java` (Metot: `yeniRafEkleme`)**
- **Açıklama: Depo kapasitesini artırmak için sisteme yeni fiziksel raflar tanımlanmasıdır.**
- **Akış:**
    1. **Kaç adet yeni raf ekleneceği sorulur.**
    2. **Döngü içerisinde her bir yeni raf için kapasite bilgisi (negatif olamaz) istenir.**
    3. **Yeni raflar listeye eklenir ve `raflar.bin` dosyasına kaydedilir.**

UC8: RAF ÇIKARMA

- **Kod Kaynağı: `RafManager.java` (Metot: `rafCikarma`)**
- **Açıklama: Depodan bir rafın silinmesi işlemidir.**
- **Güvenlik Kuralı (Veri Tutarlılığı):**
    - **Kullanıcı silmek istediği rafı seçer.**
    - **Sistem `urunManager.rafDoluMu(index)` metodu ile rafın içinde ürün olup olmadığını kontrol eder.**
    - **Eğer raf doluysa: "HATA! Raf dolu. Önce içindeki ürünleri taşıyın" uyarısı verilir ve silme işlemi kesinlikle engellenir.**
    - **Eğer raf boşsa: Raf silinir ve `urunManager.rafSilindiktenSonraGuncelle` metodu ile kayan raf numaraları düzeltilir.**

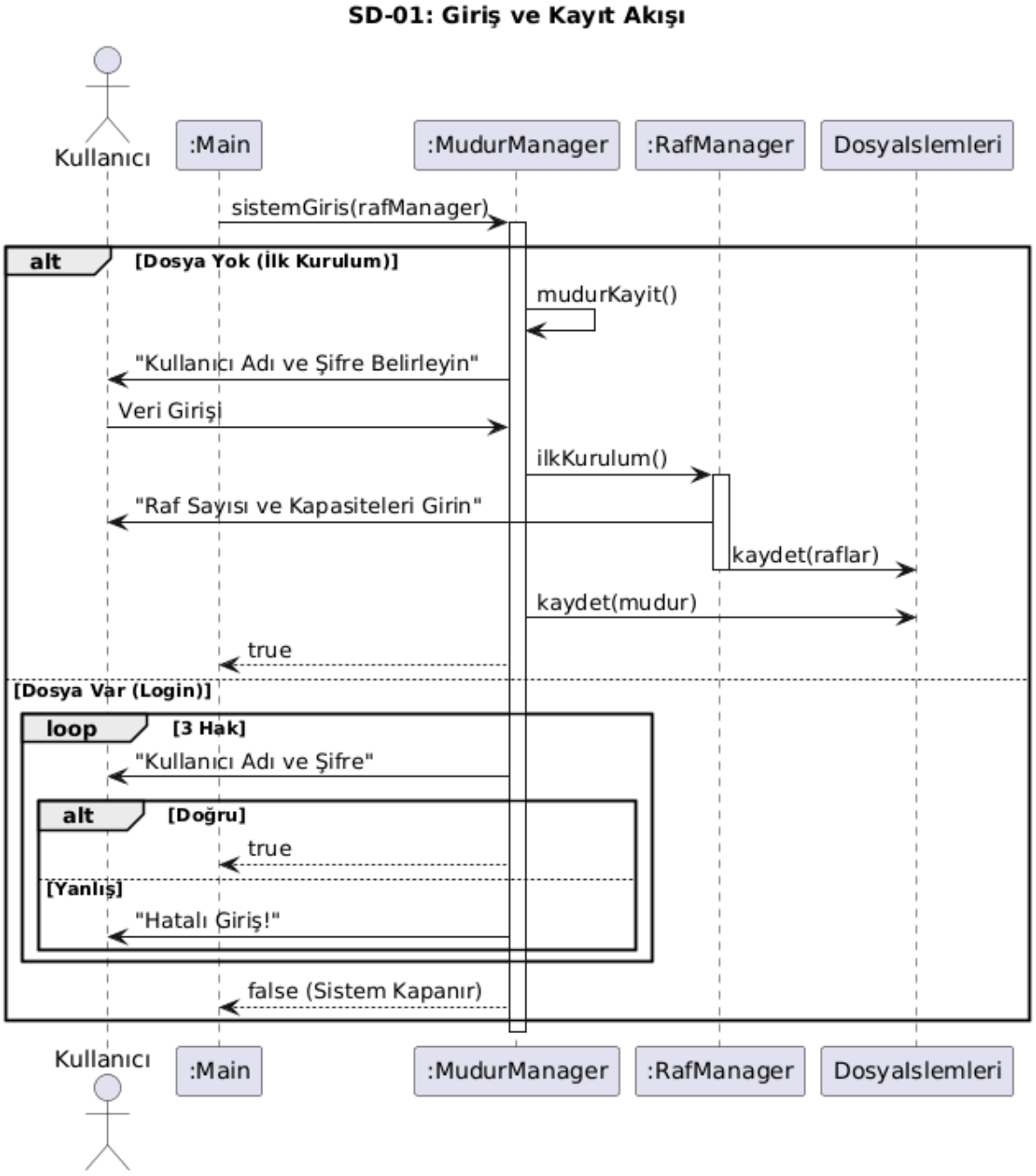
UC9: ÇIKIŞ

- **Kod Kaynağı: `Main.java` (Case 9)**
- **Açıklama: Programın güvenli bir şekilde kapatılmasıdır.**
- **Akış: Kullanıcı menüden 9'u seçtiğinde "İyi günler dileriz" mesajı görüntülenir ve `System.exit(0)` komutu ile uygulama sonlandırılır.**

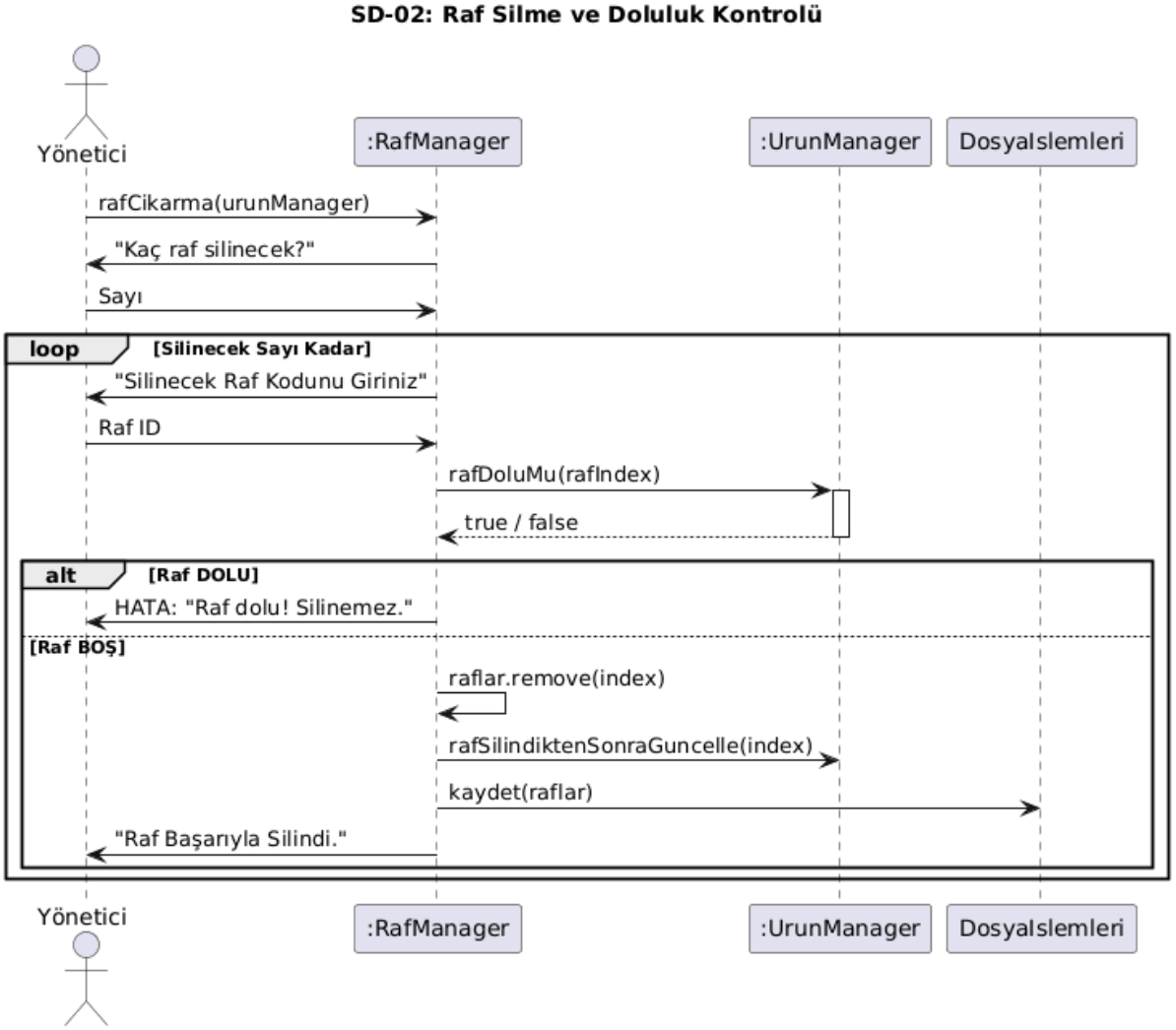
4.2 DİNAMİK MODELLER (Sequence & Activity)

Bu bölüm, sistemin en kritik algoritmalarının (Giriş Kilidi, Raf Kontrolü, Stok Dağıtımı) adım adım işleyişini gösteren Sequence diyagramlarını içerir.

4.2.1 SD-01: Güvenli Giriş ve Hata Sayacı Diyagramı *Hatalı giriş denemelerinin sayılması ve 3. hatada sistemin sonlanması sürecidir.*

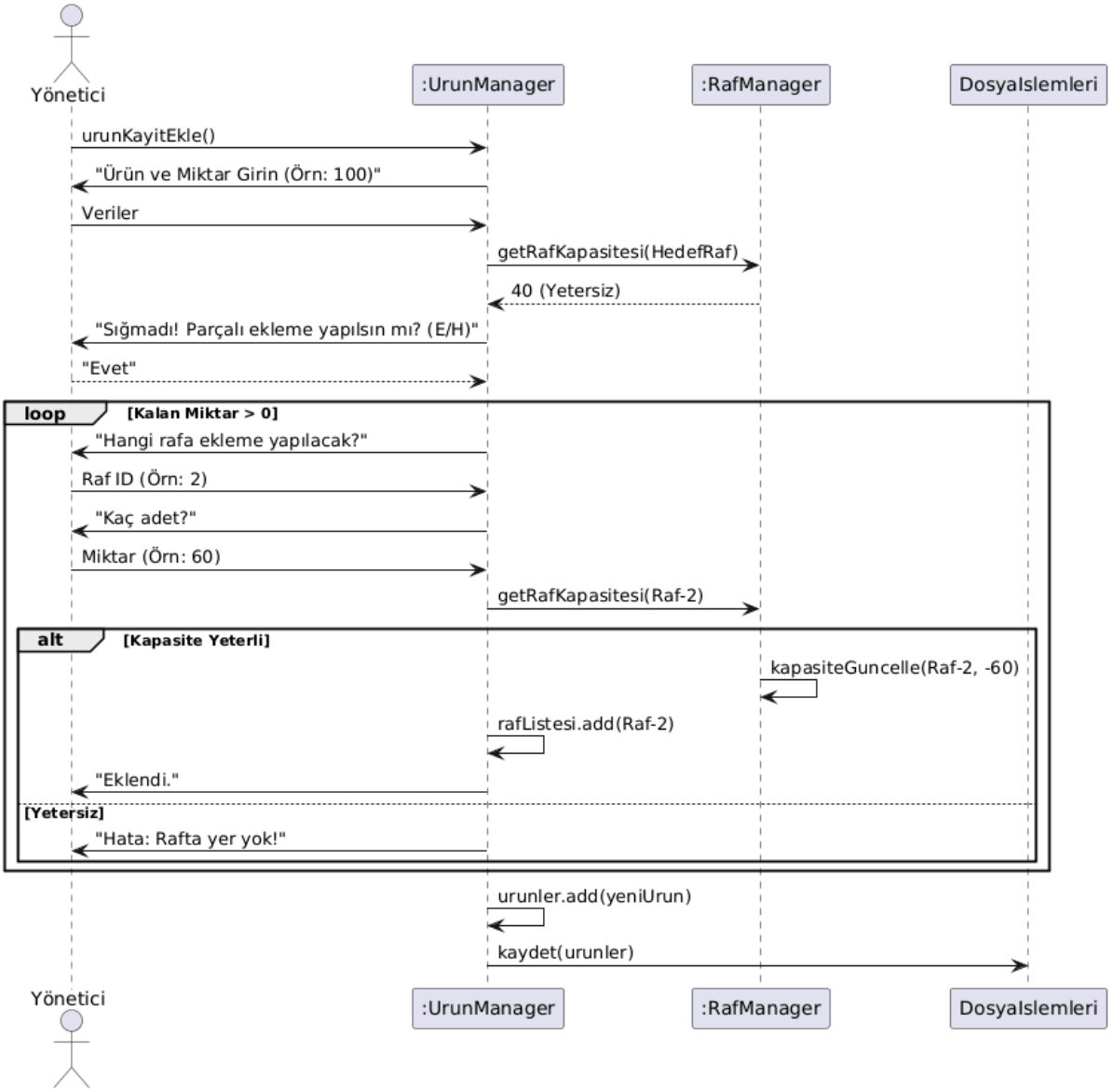


4.2.2 SD-02: Raf Silme ve Doluluk Validasyonu Diyagramı *Veri kaybını önlemek amacıyla, silinmek istenen rafın önce içerik kontrolünün yapılması sürecidir.*

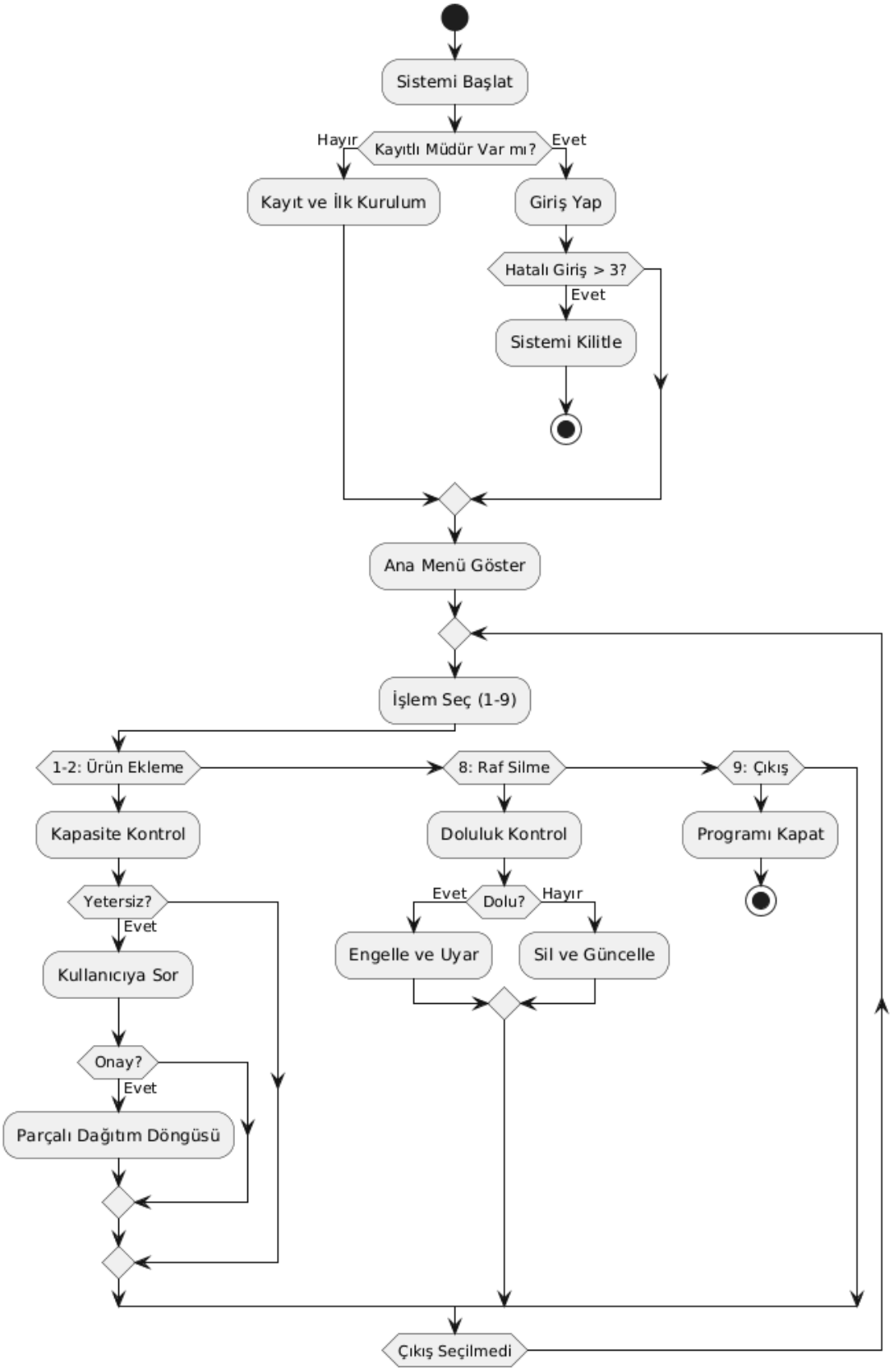


4.2.3 SD-03: Ürün Ekleme ve Akıllı Dağıtım Algoritması Diyagramı *Kapasite yetersizliğinde devreye giren "Parçalı Ekleme" algoritmasının akışıdır.*

SD-03: İnteraktif Parçalı Ürün Ekleme



4.2.4 AD-01: Sistem Genel Aktivite Diyagramı



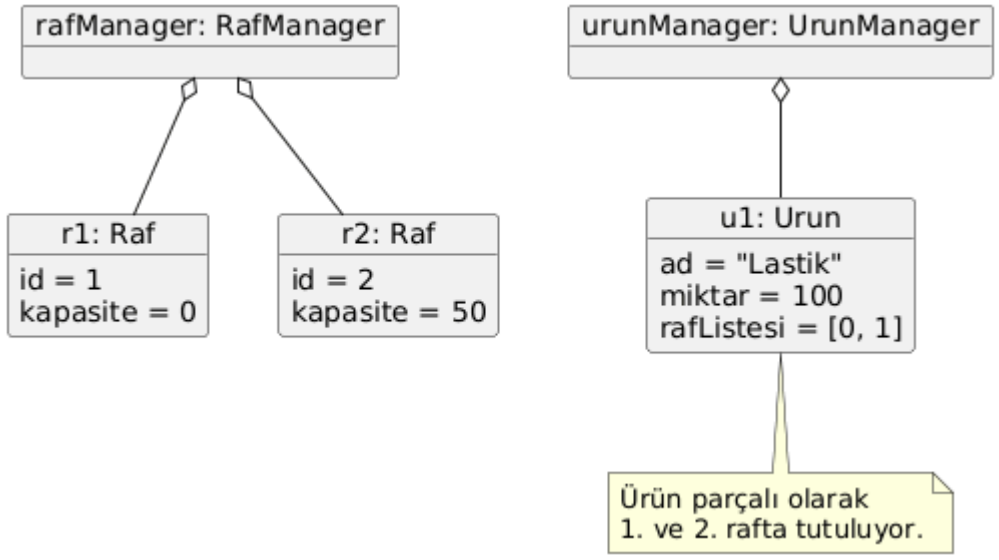
4.3 SINIF MODELLERİ (Class Analysis)

Sistem mimarisi, projenin UML şemasına uygun olarak aşağıdaki sınıflardan oluşur :

1. ***Main: Uygulamanın başlatıcısıdır.***
2. ***MudurManager: Güvenlik ve oturum yönetiminden sorumludur (`mudurGiris`, `sistemGiris`).***
3. ***UrunManager: İş mantığının merkezidir. `urunKayitEkle`, `urunTasima` ve dağıtım algoritmalarını içerir.***
4. ***RafManager: Depo fiziki yapısını yönetir (`yeniRafEkleme`, `rafCikarma`).***
5. ***Dosyalslemleri: Generic yapıda olup `kaydet` ve `yukle` metotları ile nesne serileştirme (I/O) yapar.***
6. ***Varlıklar: `Urun`, `Raf`, `Mudur`.***

4.4 NESNE MODELİ (Object Diagram)

Sistemin çalışma anındaki (Runtime) veri yapısı örneği:



5. ARAYÜZ TASARIMI VE UI AKIŞI

5.1 ANA MENÜ TASARIMI

Uygulama konsol ekranında kullanıcıya aşağıdaki işlem menüsünü sunmaktadır. Kullanıcı 1-9 arasındaki seçimlerine göre ilgili modüllere yönlendirilir.

Plaintext
##
 # DEPO YÖNETİM SİSTEMİ v1.0 #
##

1-Yeni ürün kaydı ekleme
2-Ürün ekleme
3-Ürün çıkarma
4-Ürün taşıma
5-İstenilen ürünün bilgilerini görüntüleme
6-Tüm depodaki ürünlerin bilgilerini görüntüleme
7-Depoya yeni raf ekleme
8-Depodan raf çıkarma
9-Çıkış

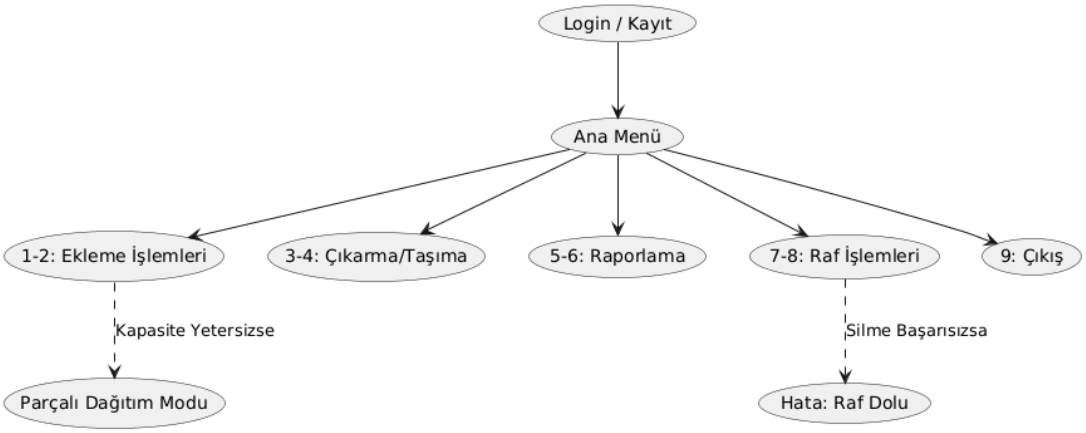
Seçiminiz: _

Tasarım Notları:

- *Ayrıştırılmış İşlemler: Kod yapısında "Ekleme" ve "Kayıt" işlemleri ayrı tutulduğu için menüde de "1-Yeni Kayıt" (hiç olmayan ürün) ve "2-Ekleme" (var olan stoğu artırma) şeklinde iki ayrı seçenek olarak sunulmuştur.*
- *Doğrudan Erişim: Alt menüler yerine, tüm kritik fonksiyonlara (taşıma, arama, raf yönetimi) ana ekrandan tek tuşla erişim sağlanmıştır.*

5.2 Kritik Uyarı Ekranları Güvenlik Kilidi: `HATA: 3 kez yanlış şifre girdiniz. SİSTEM KAPATILIYOR.` *Raf Silme Hatası:* `HATA: Raf-5 içerisinde ürün (Monitor) bulunmaktadır. SİLİNEMEZ!`

5.3 Kullanıcı Arayüzü Akış Diyagramı (UI Flow)



6. SONUÇ

Bu proje ile, 3 kişilik bir ekip olarak Java dilini kullanarak kapsamlı bir Depo Yönetim Sistemi analiz edilmiş ve tasarlanmıştır..

Analiz sürecinde;

1. ***Güvenlik: 3 hatalı girişte kilitlenen yapı ile yetkisiz erişim engellenmiştir.***
2. ***Veri Tutarlılığı: Dolu rafların silinmesini engelleyen mekanizma sayesinde veri kaybı riski ortadan kaldırılmıştır.***
3. ***Verimlilik: "Akıllı Stok Dağıtımı" algoritması sayesinde depo kapasitesi maksimum verimle kullanılmaktadır.***

Geliştirilen modeller (Use Case, Sequence, Class), projenin gerçekleştirim aşaması için sağlam bir temel oluşturmuştur.