

LAPTOP KİRALAMA SİSTEMİ - TASARIM RAPORU

1. SİSTEM MİMARİSİ ve TASARIM YAKLAŞIMI

1.1. Mimari Model: Katmanlı Mimari (Layered Architecture)

Sistem dört temel katmandan oluşur:

- Sunum Katmanı (Presentation Layer):** Kullanıcı etkileşimini yönetir
- İş Mantığı Katmanı (Business Layer):** İş kurallarını ve işlemleri uygular
- Veri Erişim Katmanı (Data Access Layer):** Veri depolama işlemlerini soyutlar
- Veri Depolama Katmanı (Data Storage):** Fiziksel veri saklama

1.2. Tasarım Desenleri ve Prensipleri

1.2.1. Singleton Deseni (FileManager Sınıfı)

- Amaç:** Uygulama genelinde tek bir dosya yöneticisi örneği
- Uygulama:** Private constructor, static instance, lazy initialization
- Fayda:** Kaynak verimliliği ve tutarlı dosya işlemleri

1.2.2. Separation of Concerns (SoC)

- Her sınıf belirli bir sorumluluğa sahiptir
- Katmanlar arası net ayırım
- Bakım ve test kolaylığı

1.2.3. Data Transfer Object (DTO) Deseni

- Customer, Laptop, Rental sınıfları veri taşıma nesneleri
- Basit veri yapıları, minimal iş mantığı
- CSV formatına kolay dönüşüm

2. VERİ YAPILARI ve DEPOLAMA TASARIMI

2.1. Dosya Tabanlı Veri Depolama

- Format:** CSV (Comma-Separated Values)
- Dosyalar:**
 - customers.txt: Müşteri kayıtları
 - laptops.txt: Laptop envanteri
 - rentals.txt: Kiralama geçmişi

- **Avantajlar:** Basit kurulum, düşük kaynak kullanımı, insan tarafından okunabilir

2.2. ID Yönetim Sistemi

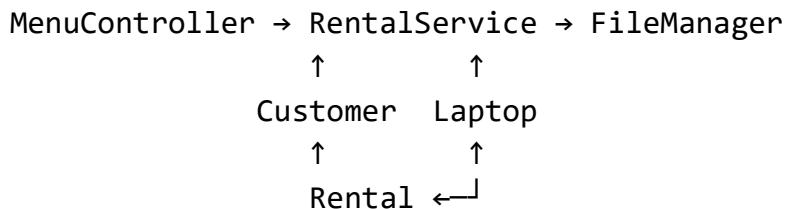
- **Yaklaşım:** Otomatik artan ID'ler
- **Mekanizma:** Her dosyanın son satırından ID okuma ve artırma
- **Başlangıç:** Boş dosya için ID = 1

2.3. Veri Bütünlüğü Mekanizmaları

- **Referans Bütünlüğü:** Rental kayıtlarında Customer ve Laptop ID referansları
- **Stok Senkronizasyonu:** Kiralama ve iadede otomatik stok güncelleme
- **Durum Yönetimi:** Rental durumu ("AKTİF"/"IADE") ile işlem kontrolü

3. SINIF İLİŞKİLERİ ve ETKİLEŞİMLER

3.1. Bağımlılık Grafiği



3.2. Akış Kontrolü

1. **Kullanıcı Girdisi** → MenuController
2. **İş Mantığı** → RentalService
3. **Veri Erişimi** → FileManager
4. **Veri Depolama** → TXT dosyaları
5. **Geri Bildirim** → MenuController → Kullanıcı

4. İŞ MANTIĞI TASARIMI

4.1. Kiralama İşlem Akışı

1. Müşteri ve Laptop ID al
2. Stok kontrolü (`Laptop.stok > 0`)
3. Tutar hesaplama (`Laptop.fiyat × gün`)
4. Stok azaltma (`Laptop.stokDusur()`)
5. Rental kaydı oluşturma

6. Dosya güncellemleri

4.2. İade İşlem Akışı

1. Kiralama ID al
2. Rental kaydı bul ve durum kontrolü ("AKTİF")
3. Durumu "IADE" olarak güncelle
4. İlgili laptop stokunu artır (Laptop.stokArtır())
5. Dosya güncellemleri

4.3. Hata Senaryoları ve Kontroller

- **Stok yetersiz:** Kiralama engelleme
- **Geçersiz ID:** Hata mesajı
- **Dosya bulunamama:** Boş liste dönüşü
- **Zaten iade edilmiş:** Durum kontrolü

5. KULLANICI ARAYÜZÜ TASARIMI

5.1. Konsol Tabanlı Menü Tasarımı

- **Yapı:** Numara tabanlı seçim sistemi
- **Navigasyon:** Sonsuz döngü menü, çıkış seçeneği
- **Girdi Temizleme:** nextLine() ile buffer temizleme

5.2. Veri Görüntüleme Formatları

- **Listeler:** Tablo formatında görüntüleme
- **Detaylar:** ID bazlı görüntüleme
- **Geçmiş:** Kiralama durumları ile listeleme

6. SİSTEM ENTEGRASYON NOKTALARI

6.1. Dahili Entegrasyonlar

- **Dosya Okuma/Yazma:** FileManager ↔ TXT dosyaları
- **Nesne Dönüşümü:** Entity ↔ CSV string
- **ID Senkronizasyonu:** Tüm entity'ler arasında tutarlılık

6.2. Harici Entegrasyon Potansiyelleri

- **Veritabanı:** TXT → SQLite/MySQL dönüşümü
- **Web Servisleri:** REST API eklenmesi
- **Raporlama:** PDF/Excel çıktıları

7. ÖLÇEKLENEBİRLİK ve GENİŞLETİLEBİRLİK

7.1. Mevcut Tasarımın Ölçeklenebilirliği

- **Güçlü Yönler:**
 - Basit mimari, kolay anlaşılır
 - Düşük kaynak gereksinimi
 - Hızlı geliştirme döngüsü
- **Sınırlamalar:**
 - Büyük veri setlerinde performans düşüşü
 - Eşzamanlı erişim desteği yok
 - Transaction yönetimi eksik

7.2. Genişletme Noktaları

1. **Yeni Entity'ler:** Yazıcı, projektör vb. kiralanabilir ekipmanlar
2. **Ek Özellikler:** İndirim sistemi, üyelik programı
3. **Raporlama:** Aylık kiralama raporları, gelir analizi

8. GÜVENLİK ve HATA DAYANIKLILIĞI

8.1. Girdi Validasyonu

- **Sayısal Girdiler:** nextInt() ile temel validasyon
- **String Girdiler:** nextLine() ile tam satır okuma
- **Stok Kontrolü:** Negatif stok önleme

8.2. Hata Yönetimi Mekanizmaları

- **Try-Catch Blokları:** Dosya işlemlerinde
- **Koşullu Kontroller:** İş mantığı seviyesinde
- **Varsayılan Davranışlar:** Dosya yoksa boş liste

8.3. Veri Kaybı Önleme

- **Tam Dosya Yazma:** Her güncellemede tüm dosyanın yeniden yazılması

- **Bellek Eşleşmesi:** RAM'deki liste ile dosya içeriği senkronizasyonu
- **Basit Backup:** Dosya kopyalama ile yedekleme potansiyeli

9. PERFORMANS TASARIM KARARLARI

9.1. Bellek Kullanımı

- **Lazy Loading:** İhtiyaç halinde dosya okuma
- **Liste Bazlı İşlemler:** Tüm verinin bellekte tutulması
- **Primitive Types:** int, double kullanımı ile performans

9.2. Zaman Karmaşıklığı

- **Okuma İşlemleri:** $O(n)$ - tüm dosyanın okunması
- **Arama İşlemleri:** $O(n)$ - lineer arama
- **Ekleme İşlemleri:** $O(1)$ - listenin sonuna ekleme
- **Güncelleme İşlemleri:** $O(n)$ - tüm listenin yeniden yazılması

10. TEST EDİLEBİLİRLİK TASARIMI

10.1. Birim Test Uygunluğu

- **Yüksek:** FileManager, Customer, Laptop, Rental sınıfları
- **Orta:** RentalService (dosya bağımlılığı)
- **Düşük:** MenuController (kullanıcı etkileşimi)

10.2. Mocking Noktaları

- **FileManager:** Dosya işlemleri mock'lanabilir
- **Scanner:** Kullanıcı girdileri mock'lanabilir
- **System.out:** Çıktılar yönlendirilebilir

11. BAKIM ve GELİŞTİRME KOLAYLIĞI

11.1. Kod Organizasyonu

- **Paket Yapısı:** Tüm sınıflar default package'ta
- **İsimlendirme:** Türkçe değişken ve metod isimleri
- **Yorum Satırları:** Minimum, anlaşılır kod öncelikli

11.2. Konfigürasyon Yönetimi

- **Hard-coded Değerler:** Dosya isimleri, durum string'leri
- **Genişletilebilirlik:** Sabitler sınıfına taşınabilir
- **Ayarlar Dosyası:** Properties file eklenebilir

12. ALTERNATİF TASARIM DEĞERLENDİRMESİ

12.1. Tercih Edilen Yaklaşımlar

1. **Dosya Tabanlı Depolama:** Basitlik ve portabilite için
2. **Singleton FileManager:** Kaynak verimliliği için
3. **CSV Formatı:** İnsan tarafından okunabilirlik için

12.2. Reddedilen Yaklaşımlar

1. **Veritabanı:** Gereksiz kompleksite
2. **Serialization:** Versioning sorunları
3. **GUI Arayüzü:** Platform bağımlılığı

13. SONUÇ ve ÖNERİLER

13.1. Tasarım Başarıları

- **Basitlik:** Anlaşılır ve sürdürülebilir
- **Modülerlik:** Katmanlar arası net ayrim
- **Esneklik:** Yeni özellik eklemeye uygun

13.2. İyileştirme Önerileri

1. **Enum Kullanımı:** "AKTİF"/"IADE" string'leri yerine enum
2. **Constant Sınıfı:** Dosya isimleri ve durumlar için
3. **Interface Tanımları:** Servis katmanı için kontratlar
4. **Logging Mekanizması:** İşlem kaydı için

13.3. Gelecek Versiyonlar için Tasarım Çıkarımları

1. **Repository Pattern:** Veri erişim soyutlaması
2. **Factory Method:** Nesne oluşturma standardizasyonu
3. **Observer Pattern:** Durum değişiklikleri için bildirimler
4. **Strategy Pattern:** Farklı fiyatlandırma stratejileri