📌 Proje Adı: Dijital Kütüphane Uygulaması

Amaç: Java Core bilgileriyle tam donanımlı bir dosya tabanlı kütüphane sistemi yazmak.

🎯 Projenin Hedefleri

- **Sınıf yapıları (POJO/DTO)**
- **OOP prensipleri (Encapsulation, Inheritance, Polymorphism)**
- **Koleksiyon yapıları (List, Map, Set)**
- **Java I/O (dosya okuma/yazma)**
- **Exception Handling**
- **Arayüzler (Interface)**
- **Enum kullanımı**
- **Basit raporlama çıktısı (System.out veya dosya)**

🧩 Projenin Parçaları

1. 🗂️ Paket Yapısı Önerisi

com.libraryapp

|

├── model // DTO'lar: Book, Member, Loan

├── service // İş mantığı

├── repository // Dosya işlemleri

├── utils // Ortak yardımcı sınıflar

└── main // Main sınıfı ve menü

📋 **Ana Menü**

--- Dijital Kütüphane ---

1. Kitap Ekle

2. Kitap Sil

3. Kitapları Listele

4. Üye Ekle

5. Kitap Ödünç Ver

6. Kitap Geri Al

7. Ödünç Kitapları Listele

8. Çıkış

🧩 **Dijital Kütüphane Uygulaması**

Yazması İçin Sınıf ve Yapı İsimleri

📁 **model Paketi – Veri Nesneleri (DTO/POJO)**

Sınıf Adı	Açıklama
Book	Kitap bilgilerini tutacak sınıf (id, başlık, yazar vb.)
Member	Üye bilgilerini tutacak sınıf (id, ad-soyad, email vb.)
Loan	Ödünç alma işlemlerini temsil edecek sınıf (kitap, üye, tarih)
BookCategory	Enum olarak kitap türlerini tutan yapı (FICTION, SCIENCE, vb.)

⚙️ **service Paketi – İş Mantığı**

Sınıf / Arayüz Adı	Açıklama
LibraryService	Arayüz: kitap ekle, sil, ödünç ver gibi temel işlemleri tanımlar
LibraryServiceImpl	LibraryService arayüzünü uygulayan, iş mantığını yazdığınız sınıf

🗂️ **repository Paketi – Kalıcı Veri / Dosya İşlemleri**

Sınıf Adı	Açıklama
BookRepository	Kitap listesini dosyaya yazan/okuyan sınıf
LoanRepository	Ödünç alma işlemlerini kalıcı hale getiren sınıf
MemberRepository	Üye bilgilerini dosyada yöneten sınıf
FileUtils	Ortak dosya okuma/yazma işlemleri burada tanımlanır

🔧 **utils Paketi – Yardımcı Araçlar**

Sınıf Adı	Açıklama
InputValidator	E-mail doğrulama, boş giriş kontrolü gibi işlemleri içerir
DateUtil	Tarih hesaplama ve biçimlendirme işlemleri
LoggerUtil	Uygulama içi loglama (opsiyonel)

🧭 **main Paketi – Uygulama Girişi**

Sınıf Adı	Açıklama
LibraryApplication	Ana menüyü ve Scanner ile kullanıcı etkileşimini yönetecek sınıf
Main	main() metodunun bulunduğu sınıf

📝 **NOT:**

Her sınıf için ;

Aşağıdaki sınıfın veri alanlarını (field) belirleyiniz,
ardından constructor, getter/setter, toString() ve gerekirse equals() gibi metodları yazınız.

Hazırlayan: Yüksek Bilgisayar Mühendisi Hamit Mızrak

✅ **Ekstra Geliştirme İçin**

- **Book ve Member sınıfları arasında ilişki var mı? Varsa nasıl kurarsınız? (id mi, nesne mi?)**
- **Loan sınıfında LocalDate kullanımı nasıl olur?**
- **LibraryServiceImpl sınıfında her işlem sonrası hangi repository çağrılmalı?**
- **Dosyaya yazarken hangi veriler hangi formatta tutulmalı? (CSV, JSON, Custom Format)**

Özellik	Açıklama
✅ Dosya üzerinden CRUD	JSON biçiminde veri saklama
✅ Enum kullanımı	Kitap kategorileri için
✅ Exception Handling	Dosya yoksa oluştur, hatalı girişlerde uyar
✅ Tarih işlemleri	LocalDate ile ödünç süreleri hesaplama
✅ Raporlama	Ödünç süresi geçen kitapları listeleme
✅ Validation	Email doğrulama, boş giriş engeli
✅ Refactoring	SOLID prensiplerine uyumlu hale getirme

📦 **Bonus Özellikler (İleri Seviye)**

- Singleton pattern ile LibraryService veya Repository
- Kitap arama (başlığa göre)
- Geçici in-memory veriler yerine dosya üzerinden veri seti başlatma
- Comparator ile kitapları yazara/gün/başlığa göre sıralama
- Logger sınıfı ile tüm işlemleri log.txt içine yazma

Hazırlayan: Yüksek Bilgisayar Mühendisi Hamit Mızrak

🧪 Test Senaryoları

1. Aynı ID ile kitap eklenirse hata ver
2. Üye olmayan kullanıcı kitap alamaz
3. Geri getirilen kitap tekrar ödünç verilebilir mi?
4. 30 günü aşan ödünçlerin listelenmesi

📘 Dijital Kütüphane Projesi – Açıklamalı Sorular

🧩 1. Proje Tasarım Soruları (OOP ve Planlama)

1. **Kütüphane sisteminizde hangi varlıklar (entity) olacak? Bunları neden seçtiniz?**
   İpucu: Kitap, Üye, Ödünç alma gibi yapıları düşünün.
2. **Her varlık için bir sınıf (class) tanımlayın. Bu sınıfların hangi özellikleri (field) olmalı?**
   Örn: Kitap için – id, başlık, yazar, kategori, uygunluk durumu.
3. **Sınıflar arası nasıl bir ilişki kurulmalı? Composition veya Association örneği kullanabilir misiniz?**
4. **Enum kullanmanız gereken bir alan var mı? (örneğin: kitap kategorileri)**
   Enum yapısı yerine String kullansaydınız ne gibi riskler olurdu?

⚙️ 2. Uygulama Soruları (Java Core)

5. **Book sınıfı için constructor, getter ve setter metotlarını yazın. Java Bean kurallarına dikkat ettiniz mi?**
6. **Loan sınıfında LocalDate kullanarak ödünç alma ve geri verme tarihlerini nasıl tanımlarsınız?**
   İpucu: LocalDate.now() ile tarih alınabilir.
7. **equals() ve hashCode() metotlarını override etmeniz gereken bir sınıf var mı? Neden?**
8. **Book nesnelerini bir List<Book> içinde saklamak istiyorsunuz. Neden ArrayList tercih ettiniz? Set kullansaydınız ne fark olurdu?**
9. **Tüm kitapları yazar adına göre sıralamak istiyorsanız hangi yapıyı kullanırsınız?**
   İpucu: Comparator<Book> ve Collections.sort().

Hazırlayan: Yüksek Bilgisayar Mühendisi Hamit Mızrak

💾 3. Dosya ve Kalıcılık (File I/O)

10. **Kitap listesini bir dosyada tutmak için hangi yapıyı kullanırsınız: BufferedWriter mi, ObjectOutputStream mi? Neden?**
11. **Program başladığında dosyada kayıtlı verileri belleğe (RAM'e) nasıl yüklersiniz? Açıklayınız.**
12. **Kitap silme işlemi yapıldığında dosyadaki veriyi güncelleme işlemi nasıl gerçekleştirilir?**
    İpucu: Dosyayı okuyup geçici bir listeye al, silinen kitabı çıkar, sonra dosyayı yeniden yaz.
13. **Dosya bulunamadığında sisteminizin çökmesini nasıl önlersiniz? Hangi istisna sınıfını kullanırsınız?**

🧰 4. İş Mantığı (Service Katmanı)

14. **Bir kitabı ödünç alma işlemini tasarlayın. Kitap başka bir kullanıcıda ise nasıl bir kontrol koyarsınız?**
15. **Aynı kitabı birden fazla kez aynı kullanıcıya ödünç verilebilmesini engelleyen bir mekanizma kurun.**
16. **Kütüphanede olmayan bir kitap ID'si girildiğinde sistem nasıl davranmalı? Exception mı? Uyarı mı?**
17. **Kitap ödünç verildikten sonra isAvailable alanını güncellemeyi unutursanız ne gibi sonuçlar doğar?**

🧪 5. Test Senaryosu ve Hata Yönetimi

18. **Bir kitap ödünç alınmışsa, tekrar ödünç verilmek istendiğinde sistem ne yapmalı?**
19. **Email adresi yanlış girilen bir üye eklenirse, sistemde bu veri nasıl yönetilmeli? Regex doğrulama kullanın.**
20. **30 günden uzun süredir geri getirilmeyen kitapları listeleyen bir fonksiyon yazın.**

🎯 6. İleri Düzey ve Opsiyonel Geliştirme

21. **LibraryService sınıfını Singleton olarak tasarlamak istiyorsanız ne yaparsınız? Singleton tasarım deseninin avantajı nedir?**
22. **Kitapları dosya yerine SQLite gibi bir veritabanına kaydetmek isterseniz, sistemde hangi sınıfları değiştirmek gerekir?**
23. **Loglama sistemi eklemek isterseniz hangi işlemleri loglamak istersiniz? Log bilgileri nereye yazılmalı?**
24. **Kullanıcı arayüzü olarak Scanner ile CLI menü kullandınız. Bunun yerine Swing veya JavaFX arayüzü yapsaydınız ne gibi avantajlar sağlardı?**

✅ Uygulama Sonrası Tartışma Soruları

- Bu projeyi geliştirirken en çok zorlandığınız konu neydi?
- Daha büyük çapta bir uygulamaya evriltmek isterseniz neleri modüler hale getirmelisiniz?
- Kod tekrarını azaltmak için nelere dikkat ettiniz?
- SOLID prensiplerinden hangilerini uygulayabildiniz?

Hazırlayan: Yüksek Bilgisayar Mühendisi Hamit Mızrak

1. 🚀 Genel Amaç

Bu proje, öğrendiğiniz konuları barındıran bir java core projesidir. **Java programlama diline ait temel kavramları** gerçek hayat senaryoları üzerinden uygulamalı olarak öğrenmesini sağlamayı amaçlar. Amaç, salt teorik bilgileri değil, bu bilgilerin **nasıl yazılıma dönüştüğünü anlamaktır.**

2. 🧠 Eğitsel Amaçlar

Amaç No	Açıklama
EA1	Nesne Yönelimli Programlama (OOP) yaklaşımını gerçek dünya nesneleri üzerinden uygulama: Kitap, Üye, Ödünç alma vb.
EA2	Veri yapıları (List, Map, Set) ile dinamik ve esnek veri yönetimi gerçekleştirme
EA3	Java Core konularını (class, object, interface, enum, exception, file I/O vb.) entegre bir şekilde kullanma becerisi kazanma
EA4	Dosya işlemleri (FileReader, BufferedWriter, FileWriter vb.) ile kalıcı veri tutmayı öğrenme
EA5	Scanner, LocalDate, Collections, Comparator gibi temel sınıflarla kod yazma pratiği geliştirme
EA6	Temiz, modüler ve okunabilir kod yazma alışkanlığı kazanma (SOLID ilkelerine giriş düzeyinde uyum)

Hazırlayan: Yüksek Bilgisayar Mühendisi Hamit Mızrak

3. 👷 Teknik Amaçlar

Amaç No	Açıklama
TA1	model-service-repository-main katmanlarına ayrılmış bir uygulama mimarisi inşa etmek
TA2	Gerçek bir sistemdeki işlemleri (ekleme, silme, arama, listeleme) dosyaya dayalı olarak simüle etmek
TA3	Exception handling yapısıyla hata yönetimi bilinci geliştirmek
TA4	Enum tipiyle sabit verilerin (örneğin: Kitap kategorileri) daha güvenli ve kontrollü kullanılmasını sağlamak
TA5	CRUD işlemlerini kullanıcıdan alınan verilerle ve kalıcı ortamla birleştirmek

4. 🛠️ Becerisel/Kazanımsal Amaçlar

Amaç No	Açıklama
KA1	Gerçek dünya problemini analiz edip yazılım sistemine modelleme yetisi kazanma
KA2	Sınıflar ve nesneler arası ilişkileri analiz etme (composition, dependency)
KA3	Projeye sıfırdan başlama, geliştirme ve test etme süreçlerini uçtan uca deneyimleme
KA4	Temel bir menü sistemini (CLI) kurarak kullanıcı arayüzü oluşturma
KA5	Raporlama mantığını dosyaya veya ekrana yansıtarak veri analizi yapabilme (örneğin: en çok ödünç alınan kitaplar, geciken iadeler vb.)
KA6	Kodları test etme, senaryolara göre davranışları gözlemleme ve debug yapabilme

5. 📈 Proje Sonrası Kazanımlar

- Gerçek bir uygulamanın sıfırdan geliştirildiği tüm aşamaları deneyimlemiş olacaklar.
- Kodların sadece yazılması değil, **planlanması, test edilmesi, yönetilmesi** süreçlerini yaşayacaklar.
- Temel Java bilgilerini soyuttan çıkarıp somut uygulamaya dönüştürme pratiği kazanacaklar.
- “Sadece çalışan kod değil, sürdürülebilir, genişletilebilir kod” yaklaşımına ilk adım atılmış olacak.