Genel Özet: Bu projede yeni kurulan bir bankanın otomasyonunu simüle ettik. Kodun amacı, bazı özelliklere sahip hesaplar oluşturup üzerinde işlemler yapmasına, bakiye yatırmasına ve çekmesine, faiz hesaplamasına ve işlem geçmişini görüntülemesine olanak tanımaktır. Benim hazırladığım Java kodunda, kullanıcıdan alınan ID ve başlangıç bakiye bilgisini alarak hesap oluşturuyoruz ve bu hesabın bilgilerini "user.txt" dosyasında tutuyoruz. Ayrıca, bu hesabın yaptığı işlemleri kaydetmek için "hesapIDsi.txt" adında bir dosya oluşturuyoruz ve bu hesap için yaptığımız her işlemde gerekli bilgileri bu dosyaya kaydediyoruz. Bazı işlemlerde bu bilgileri çekip işleyip değiştiriyoruz. Program gerçek zamanlı ilerlemektedir; sistemde varsayılan tarih ve aktif zamanlı tarih bulunmaktadır, dosyaların kayıt ve işlem tarihleri aktif tarihle tutulmaktadır. Son olarak, menümüzde kullanıcılarI bu işlemleri kullanabildikleri bir menü ile karşılıyoruz.  
  
  
Sınıfları tanımlama:

Toplam 7 adet sınıf içermektedir.

1. `Account` Sınıfı:

- Bu sınıf, banka hesabının temel özelliklerini ve işlevlerini tanımlayan soyut bir sınıftır.

- Hesap ID'si, bakiye, başlangıç bakiyesi, günlük faiz oranı gibi bilgileri içerir.

- Yatırma, çekme, faiz hesaplama gibi işlemleri gerçekleştirmek için yöntemler sağlar.

- Bu sınıftan üretilen her nesne bir banka hesabıdır.

Account sınıfı Metod tanımları:

- Account(): Parametresiz bir yapıcı metod. Sistem varsayılan tarihini ve hesap tarihini ayarlar.

- Account(int id, float initialBalance, String hesaptur): İki parametre alır (id ve başlangıç bakiyesi). Hesap türünü belirler ve hesap tarihini ayarlar.

- Account(int id, float balance, float initialBalance, String hesaptur, LocalDate account\_date): Beş parametre alır (id, bakiye, başlangıç bakiyesi, hesap türü ve hesap tarihi). Bu parametreleri kullanarak hesap nesnesini oluşturur.

- getId(): Hesap ID'sini döndürür.

- getBalance(): Hesap bakiyesini döndürür.

- getInitialBalance(): Başlangıç bakiyesini döndürür.

- getHesaptur(): Hesap türünü döndürür.

- getAccountDate(): Hesap tarihini döndürür.

- tarih(): Sistem başlangıç tarihini yazdırır.

- deposit(float amount): Belirli bir tutarda para yatırma işlemini gerçekleştirir.

- transactionHistory(String islem): İşlem geçmişine bir işlemi ve tarihini ekler.

- benefit(): Faiz hesaplama işlemlerini yapmak için soyut bir metod.

- showWithdraw(float balance, float initialBalance): Para çekme işlemini gerçekleştirmek için soyut bir metod.

- withdraw(float cesh): Belirli bir tutarda para çekme işlemini gerçekleştirir.

- systemDate(int yil, int ay, int gun): Sistemin geçerli tarihini belirler.

- systemDatenow(): Sistemin geçerli tarihini bugünün tarihi olarak ayarlar.

2. `ShortTermAccount` Sınıfı:

- `Account` sınıfından türetilmiş bir alt sınıftır.

- Kısa vadeli hesap türünü temsil eder.

- Kendi kurallarına göre yatırma ve çekme işlemlerini kontrol eder.

- Günlük faiz oranını hesaplamak için `benefit` yöntemini uygular.

ShortTermAccount sınıfı Metod tanımları:

- deposit(float balance): Belirli bir tutarda para yatırma işlemini gerçekleştirir.

- showWithdraw(float balance, float initialBalance): Para çekme işlemini gerçekleştirmek için gerekli kontrolleri yapar.

- benefit(): ShortTerm hesabının günlük faizini hesaplar.

3. `LongTermAccount` Sınıfı:

- `Account` sınıfından türetilmiş bir alt sınıftır.

- Uzun vadeli hesap türünü temsil eder.

- Kendi kurallarına göre yatırma ve çekme işlemlerini kontrol eder.

- Günlük faiz oranını hesaplamak için `benefit` yöntemini uygular.

LongTermAccount sınıfı Metod tanımları:

- deposit(float balance): Belirli bir tutarda para yatırma işlemini gerçekleştirir.

- showWithdraw(float balance, float initialBalance): Para çekme işlemini gerçekleştirmek için gerekli kontrolleri yapar.

- benefit(): LongTerm hesabının günlük faizini hesaplar.

4. `SpecialAccount` Sınıfı:

- `Account` sınıfından türetilmiş bir alt sınıftır.

- Özel hesap türünü temsil eder.

- Kendi kurallarına göre yatırma ve çekme işlemlerini kontrol eder.

- Günlük faiz oranını hesaplamak için `benefit` yöntemini uygular.

SpecialAccount sınıfı Metod tanımları:

- deposit(float balance): Belirli bir tutarda para yatırma işlemini gerçekleştirir.

- showWithdraw(float balance, float initialBalance): Para çekme işlemini gerçekleştirmek için gerekli kontrolleri yapar.

- benefit(): Special türü hesabının günlük faizini hesaplar.

5. `CurrentAccount` Sınıfı:

- `Account` sınıfından türetilmiş bir alt sınıftır.

- Vadesiz hesap türünü temsil eder.

- Yatırma ve çekme işlemlerini kontrol eder.

- Günlük faiz oranı sıfırdır.

CurrentAccount sınıfı Metod tanımları:

- deposit(float balance): Belirli bir tutarda para yatırma işlemini gerçekleştirir.

- showWithdraw(float balance, float initialBalance): Para çekme işlemini gerçekleştirmek için gerekli kontrolleri yapar.

- benefit(): günlük faiz değerini 0 olarak döndürür;

6. `Bank` Sınıfı:

- `Account` sınıfından türetilmiş bir alt sınıftır.

- Banka hesabı oluşturma ve işlem yapma işlevlerini içerir.

- Hesapları dosyalara yazma ve okuma işlemlerini gerçekleştirir.

- Hesap ID'leri oluşturur ve hesap bilgilerini elde etmek için gerekli yöntemleri sağlar.

-Özel hesaplar arasında çekiliş yapma fonksionunu içerir.

Bank sınıfı metod tanımları:

- `showWithdraw(float balance, float initialBalance)`: bu metodun super calssta abstact olarak tanımlı olduğu için içi boştur ve herhangi bir işlevsellik sağlamaz.

- `benefit()`: bu metodun super calssta abstact olarak tanımlı olduğu için içi boştur ve herhangi bir işlevsellik sağlamaz.

- `Bank()`: Bu, parametresiz bir kurucu metoddur. Ancak, herhangi bir işlevsellik sağlamaz.

- `AccountCreater(int op) throws IOException`: Bu metod, kullanıcıdan girdiler alarak yeni bir hesap oluşturur. `op` parametresi, kullanıcının hangi tür hesap açmak istediğini belirtir. `op` değerine göre kullanıcıdan gerekli bilgiler alınır ve ilgili hesap türüne ait bir nesne oluşturulur. Oluşturulan hesap nesnesi `printAccount` metoduna gönderilir ve kullanıcının kaydı tamamlanır.

- `printAccount(Account account) throws IOException`: Bu metod, bir hesabın bilgilerini kullanıcı dosyasına yazan ve ayrıca hesap için bir dosya oluşturan işlemleri gerçekleştirir.

- `generateAccountID(int id) throws IOException`: Bu metod, bir hesap için bir dosya oluşturur. Hesap kimliği (`id`) parametresiyle çağrılır ve kimliği kullanarak bir dosya adı oluşturur.

- `Accountinquiry(int op) throws IOException`: Bu metod, kullanıcıya ait bir hesabın kimlik bilgisini alır. `op` parametresi, kullanıcının hesap sorgulama veya yeni bir hesap oluşturma amacıyla bu metodu kullandığını belirtir. Kullanıcıdan bir hesap kimliği alır, bu kimliği kullanarak ilgili dosyayı kontrol eder ve uygun mesajları kullanıcıya gönderir.

- `deposit() throws IOException`: Bu metod, para yatırma işlemi için kullanılır. Kullanıcıdan bir hesap kimliği ve yatırılacak miktarı alır. Hesap bilgileri okunur, ilgili hesap nesnesi oluşturulur ve para yatırma işlemi gerçekleştirilir. Hesap bilgileri ve işlem geçmişi güncellenir.

- `getAccountInfo(int id) throws IOException`: Bu metod, bir hesap hakkında bilgileri kullanıcı dosyasından alır. Hesap kimliği (`id`) parametresiyle çağrılır, ilgili dosyayı okur ve hesap bilgilerini bir dize olarak döndürür.

Tabii, aşağıda `Bank` sınıfının metodlarının amaçlarını tanımlıyorum:

- `withdraw()`: Kullanıcıdan bir hesap ID'si alır, bu ID'ye sahip hesabın bilgilerini alır ve hesap türüne göre ilgili hesap sınıfını oluşturur. Ardından, hesaptan para çekme işlemini gerçekleştirir, bilgileri günceller ve işlem geçmişini yazdırır.

- `updateInformation(int id)`: Verilen bir hesap ID'sine göre kullanıcı bilgilerini günceller. `user.txt` dosyasından hesap ID'sini içeren satırı bulur, bu satırı geçici bir dosyaya yazmadan atlar ve diğer satırları geçici dosyaya kopyalar. Ardından, orijinal `user.txt` dosyasını siler ve geçici dosyanın adını `user.txt` olarak değiştirir.

- `ShowIDs()`: Tüm hesap ID'lerini `user.txt` dosyasından okur ve ekrana yazdırır.

- `transactionHistoryprint(int id, String str)`: Verilen bir hesap ID'sine ve işlem bilgisine dayanarak, ilgili hesap için işlem geçmişine satır ekler. Hesap ID'sini kullanarak bir dosya adı oluşturur ve bu dosyaya işlem bilgisini yazar.

- `transactionHistoryShow()`: Kullanıcının belirli bir hesap için işlem geçmişini görüntüler. Hesap ID'sini kullanarak ilgili dosyayı okur ve en son 5 işlemi ekrana yazdırır.

- `systemDateCreat()`: Kullanıcıya bugünkü tarih veya özel bir tarih girmesini sağlar. Kullanıcının seçimine göre `systemDatenow()` veya `systemDate(yil, ay, gun)` metodlarını çağırır.

- `AllSpecialAccounts()`: `user.txt` dosyasından tüm özel hesapları (SpecialAccount) okur ve hesap ID'si ve bakiyesini içeren bir liste döndürür.

- `Sortition()`: Tüm özel hesapları alır ve bu hesaplardan rastgele bir hesap seçer. Seçilen hesaba 1000 TL ödül verir, işlem geçmişine bu bilgiyi yazar, hesabın bakiyesini 1000 TL artırır ve bilgileri günceller.

7. `Main` Sınıfı:

-Bu sınıf sadece Bank turunden bir nesne kullanıcıya islemlerini yapmasını sağlar.

-Kulanıcıya arayüz olanağı sağlar.

Main sınıfı metod tanımları:

- `main(String[] args)`: Programın başlangıç noktasıdır. `Bank` sınıfından bir örnek oluşturur ve `menu()` metodunu çağırarak ana menüyü başlatır.

- `menu(Bank hs)`: Kullanıcı arayüzünü sunan ana menüyü oluşturur. Kullanıcıdan bir seçenek girmesini ister ve seçeneğe göre ilgili `Bank` sınıfı metodunu çağırır. Kullanıcı çıkış yapana kadar menüyü tekrar tekrar gösterir.

Bu sınıflar, farklı banka hesap türlerini temsil eder ve her biri hesap işlemlerini yönetmek için gerekli yöntemleri sağlar.

Uml diyahramının .uml ve .png uzantılı dosyasını rapora ekledim ordan daha net bir şekilde incelenebilmektedir.





Eğer id kayıtlı ise veya minimum bakiyenin altında bir değer girersek uyarı mesajıyla birlikte bilgiyi tekrar istiyor. Bu özellik diğer hesaplar içinde geçerli.





Para ekleme yönteminde, eğer ID bulunamazsa kullanıcı kayıtlı değil bilgisini ekrana yazdırıyor. Eğer ekleyeceğiniz para değeri mantık dışıysa hata mesajıyla tekrar bilgi isteniyor. Son olarak, eğer tüm işlemler doğruysa işlem başarılı yazısı geliyor. Aynı durum para çekme işlemi için de geçerlidir.











