Techno Study Java Project 3

Proje Konusu: Banka Uygulaması

1. Customer (Müşteri) Sınıfı

- Fields: customerId (int), firstName (String), lastName (String), city (String)
- Cunstructors: Customer(), Customer(int customerId, String firstName, String lastName, String city)
- Getter ve Setter metodları
- toString() metodu

2. AccountType (Hesap Tipi) Enum Sınıfı

- Sabitler: CHECKING (Vadesiz Hesap), SAVINGS (Tasarruf Hesabı), CREDIT(Kredi Hesabı)

3. Account (Hesaplar) Sınıfı

- Fields: accountId (int), customerId (int), balance (double), accountType (AccountType)
- Constructors: Account(){}, Account(int accountId, int customerId, AccountType accountType){}
- Getter ve Setter metodları
- depositToAmount(double amount) metodu
- withdrawToAmount(double amount) metodu
- toString() metodu

4. BankSimulation Sınıfı

- Fields: customers (Map<Integer, Customer>), accounts (List<Account>), scanner (Scanner)

- Constructors: BankSimulation(){},

```
BankSimulation() {
customers = new HashMap<>();
accounts = new ArrayList<>();
scanner = new Scanner(System.in);
}
```

- Ana metot: public static void main(String[] args)
- run() metodu
- listCustomers() metodu

- listCustomerAccounts(Customer customer) metodu
- addCustomer() metodu
- customerOperationsMenu() metodu
- customerOperations(Customer customer) metodu
- openNewAccount(Customer customer) metodu
- depositToAccount(Customer customer) metodu
- withdrawToAccount(Customer customer) metodu
- checkBalance(Customer customer) metodu
- getAccountById(int accountId) metodu

Proje Yapısı:

```
BankingSystem
    ├── src
    |   ├── Customer.java
    |   ├── AccountType.java
    |   ├── Account.java
    |   └── BankSimulation.java
    └── README.txt (İsteğe Bağlı)
```

Görevler

1. Proje yapısını oluşturun ve verilen sınıfları ilgili dosyalara yerleştirin.
2. "Customer" sınıfına getter ve setter metodları ekleyin.
3. "Account" sınıfına "depositToAmount()" ve "withdrawToAmount()" metodlarını ekleyin.
4. "BankSimulation" sınıfında müşteri ekleme ve listeleme fonksiyonlarını tamamlayın.
5. "BankSimulation" sınıfında yeni hesap açma ve hesap işlemleri fonksiyonlarını tamamlayın.
6. Main metodu içinde uygulamayı başlatın ve kullanıcı arayüzü ile işlemleri gerçekleştirin.

Notlar

- Proje sırasında hatalı girişlere karşı uygun hata yönetimi isteğe bağlıdır..

EK BİLGİLER:

`BankSimulation` sınıfı, banka işlemlerini yöneten ve kullanıcı arayüzü sağlayan ana sınıftır. `main` metodu bu sınıfta yer alır ve banka simulasyonunu başlatır. Aynı zamanda `run` metodu ana menüyü oluşturur ve kullanıcıdan girdiler alarak işlemleri yönlendirir. `BankSimulation` sınıfı içinde müşterileri ve hesapları temsil eden maps ve liste şeklinde veri yapıları (customers ve accounts) kullanılmaktadır.

`Customer` sınıfı, bankadaki müşterileri temsil eder ve müşterilere ait bilgileri içerir. Daha doğrusu, bir Customer nesnesi oluşturduğumuzda, bu nesne vasıtasıyla Customer sınıfının fieldlerini (değişkenlerini) ve metodlarını tutacak ve etkileyecek hale gelmiş oluruz.

`Account` sınıfı, bankadaki hesapları temsil eder ve hesaplara ait bilgileri içerir. Daha doğrusu Account cinsinden oluşturduğumuz nesne, hesap bilgilerini ve metodlarını tutmamızı ve onları kullanmamızı sağlar... Account sınıfı`withdrawToAmount` (para çek), `depositToAmount` gibi işlem metotlarını içerir.

`AccountType` enum, vadeli-vadesiz ve kredi hesabı türlerini temsil eder.

`BankSimulation` sınıfı içindeki `accounts` ve `customers` değişkenleri `final` olmasının nedeni, bu değişkenlere atanan referansların sonradan değiştirilmesini engellemek ve programın daha güvenli ve kontrol edilebilir olmasını sağlamaktır. `final` anahtar kelimesiyle tanımlanan bir değişkenin değeri sadece bir kez atanabilir ve sonradan değiştirilemez.
Örneğin, `customers` değişkeni için:

`private final Map<Integer, Customer> customers;`

Bu tanımlama, `customers` değişkeninin referansının değiştirilemeyeceği anlamına gelir. Ancak, `customers` değişkenine atanan `HashMap` nesnesinin içeriği yine de değiştirilebilir. Yani, `customers` değişkeninin referansı aynı kalacak şekilde müşteri bilgileri eklenebilir, güncellenebilir veya silinebilir.