**Nesneye Yönelik Programlama (NYP) Proje Raporu:**

**MOUSSA BANE 20360859102**

Bu projeyi başlamadan önce verilen dokümanı inceledim, neye hangi değişkenlere ihtiyacım olacağı listeleyebilmek için.

Proje 8 tane sınıftan oluşmaktadır. **JA33\_NYP\_PROJECT** adlı sınıfı **JFrame** ata sınıfından miras alan ve projenin main metodu içeren bir sınıftır. **Kisi** adlı sınıf **BankaPersonli** ve **Musteri** adlı sınıfların ata sınıfıdır; yani o iki sınıf **Kisi** ata sınıfından miras almaktadır. **KrediKarti** adlı sınıf hiç sınıftan miras almıyor. Son olarak **BankaHesabi** adlı sınıf **VadesizHesap** ve **YatirimHesabi** adlı sınıfların ata sınıfıdır.

Projeme başlamak için madem **KrediKarti** hiç sınıftan miras almıyor, o sınıfı ile başladım. Sınıfın private değişkenlerin referans değişkenleri belirledim ve 2 tane double tipinde parametreli(limit ve guncelBorc) constructor’u belirledim. Constructor’un içinde bu değişkenlere atamalar yapıldı ve kartNumarasi Math.random() kullanarak rasgele sayı yaratıp ve kartNumarasi’na atama yapıldı. KullanilabilirLimit ise limit ve guncelBorc toplamı olarak tanımladım. Madem private olarak tanımlanan değişkenlerine bu sınıfın dışında erişilemez her biri için gerekli **set** ve **get** metotları yazdım.  **get** metotlar geri döndüren değer olan ve parametre almayan metotlardır. **set** metotları ise geri döndüren değer olmayan ve parametre alan metotlardır, parametre olarak geçen değişken veya nesnenin üzerinde değişlik yapabilmemizi sağlar. Bu sınıfın sonu olarak, sınıfın bilgileri string olarak gösterebilmemiz için toString() onun toString() metodu yazdım, java’da Object ata sınıfından override olan bir metottur.

Ardından **BankaHesabi** sınıfına geçtim. Sınıfın private olan değişkenlerin referans değişkeni belirleyip parametreli constructor’un içerisinde onlara atamalar yaptım. Iban değişkeni Math.random kullanarak rasgele sağı türetilip “TR” string’e eklenir ve atama yapılır. Iban değişkenin **set** metodu gerekmez çünkü otomatik olarak oluşturur sadece onun **get** metodu yazdım. Ve diğer private değişkenin **set** ve **get** metotları yazdım. Bu sınıfın sonu olarak, içerisindeki bilgileri tek bir string olarak elde edebilmesi için onun toString() fonksiyonu da yaptım.

Ondan sonra, **BankaHesabi** ata sınıfından miras alan sınıflarına geçtim. Önce **VadesizHasap** sınıfı, private olarak hesap türü tutan string tipinde hesapTuru değişkeni tanımladım. Sonra da onun **set** ve **get** metotları ve constructor’u yaptım.

***public void paraTransferi(BankaHesabi aliciHesap,BankaHesabi gonderenHesap,double miktar )*** fonksiyonu para transferleri gerçekleştirir. Parametre olarak **BankaHesabi** tipinde alıcı hesap , gönderen hesabı ve double tipinde gönderilecek miktarı alınmaktadır. Bu fonksiyon içindeki işlemler, para transferi yapıldıktan sonra gönderen hesabın bakiyesi miktar kadar azalırken alıcı hesabın bakiyesi miktar kadar yükselmekte, gerçekleştirir. ***public void krediKartiBorcOdeme(KrediKarti krediKarti,double odenecekMiktar)*** metodu ise kredi kartı borç ödeme işlemi gerçekleştirir. Parametre olarak ilgili kredi kartı ve ödenecek miktarı alınır. İşlemi gerçekleştirildikten sonra bu kartın güncel borcu miktar kadar azalmakta ve güncel borç azalırsa kullanılabilir limiti de miktar kadar azalmaktadır. Bu sınıfın sonu olarak **BankaHesabi** ata sınıfından *toString*() override yaptım. Sonra **YatirimHesabi** sınıfına geçtim. Bu sınıfın içerisinde de private olarak hesap türü tutan string tipinde hesapTuru değişkeni tanımladım. Ardından onun **set** ve **get** metotları ve sınıfın constructor’u yaptım. ***public void paraEkle(BankaHesabi hesap,double eklenecekMiktar)*** metodu kullanarak ilgili hesaba para ekleme işlemi gerçekleşir. Parametre olarak hangi hesabına para eklemek istersek onu belirliyoruz ve ekelenecek miktarı da. Bu metodu çalıştırdıktan sonra ilgili hesabın bakiyesi miktar kadar yükselmektedir. Bunu yanı sıra ***public void paraCek(BankaHesabi hesap,double cekilecekMiktar)*** para çekme işlemi gerçekleştirebilen metodumuz da var. Diğerinde fark olarak çekilecek miktar kadar ilgili hesabın bakiyesi miktar kadar azalmaktadır. Bu sınıfın sonu olarak da **BankaHesabi** ata sınıfından *toString*() override yaptım.

Ardından **Kisi** sınıfına geçtim. Bu sınıf hiç sınıftan miras almıyor. **BankaPersoneli** ve **Musteri** sınıfların ata sınıfıdır. Private olarak belirlenen referans değişkenleri(ad , soyad , email ve telefonNumarası ) belirledim ve parametreli constructor’un içerisinde onlar atamlar yapıldı. Bu değişkenlerin **set** ve **get** metotları da yaptım. Sınıfın sonu olarak toString()’i yaptım. Sonra **BankaPersoneli** , **Kisi**

ata sınıfından miras alan sınıfa geçtim. Müşteri tutan ArrayList tipinde olan müşteriler adlı referans değişkeni ve personelID belirledim. Constructor’un içinde müşteriler tutan yeni bir ArrayList tanımlandım ve Math.random

kullanarak rasgele sayı üretip personlID’ye aktarılmaktadır. MusteriEkle ve MusteriSil metotların sayesinde ilgili müşteri parametre olarak verip o müşteriyi ekleyebiliriz veya silebiliriz. Ardından müşteriler göster diye bir fonksiyon tanımladım, müşteriler listesinde olan müşteriler string olarak döndürebilmektedir. Bu sınıfın sonu olarak toString()’i de yaptım. Ardından **Musteri** adlı sınıfı **Kisi** ata sınıfından miras almaktadır. Constructor’unda musteriNumarası Math.radom kullanarak tanımladım ve müşterinin hesapları ve kredi kartları tutan 2 tane yeni ArrayList tanımladım. MusteriNumarasının **get** metodu da yaptım. ***public void hesapEkle(BankaHesabi yeniHesap,double bakiye)*** metodu kullanarak hesap türüne göre yeni bir hesap oluşturup hasaplar ArrayList'ine eklenir. ***public void hesapSil(BankaHesabi silinecekHesap)*** metodu kullanarak silmek istediğimiz hesabın bakiyesine bağlı olarak silinir veya konsola bir uyarı verilmektedir(hesabın bakiyesi 0’a eşikse silinsin değilse konsola bir uyarı verilsin). ***public void krediKartiEkle(KrediKarti krediKarti)*** metodu kullanarak kredi kartı ekleme işlemi gerçekleştirilir. ***public void krediKartiSil(KrediKarti silinecekKart)*** metodun sayesinde silmek istediğimiz krediKartin güncelBorca bağlı olarak silinip veya konsola bir uyarı yazılmaktadır(kredi kartın güncel borcu 0’a eşitse silinsin değilse konsola bir uyarı yazılsın). ***public String getHesaplar()*** metodun sayesinde hesapList'imizi string olarak elimizde olacaktır. ***public String getKrediKartlar()*** metodun sayesinde hesaplarla yaptığım aynı şey, burada kartlar içindir. Ve sınıfın sonu olarak toString()’i ata sınıfından override metodu yaptım.

Projemin sonu olarak **JA33\_NYP\_PROJECT** main metodu olan ve JFrame ata sınıfından miras alan sınıfa geçtim. Pencerem için 4 **JButton**, 7 **JTextField** ve 6 **JLabel** tanımladım. Ardından gerekli ayarlamalar yaptım kodda gösterdiğim gibi. Pozisyonla ilgili bazı component’ler için pencerenin boyu ve genişliğine bağlı olarak değişsin diye **this.getWidth()** ve **this.getHeight()** kullandım. Sonra ***public void actionPerformed(ActionEvent actionEvent)*** metoduna ActionListener interface'inden override yaptım. Basıldığı butona bağlı olarak işlemler gerçekleştirilmektedir. “Programım çalıştırdığımda fark ettim ki textFiel içindeki text alt satıra geçme olmaz”.

