Ahmet Mert Gülbahçe
141044015

CSE443

HW3

PART 1

Bu bölümde, benzer görevi yerine getiren iki tane arayüzümüz bulunmaktadır.

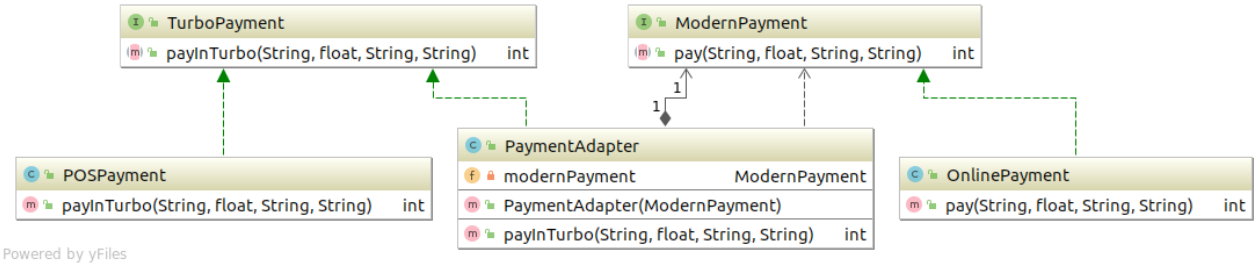
`TurboPayment`: 1980'lerden kalma eski bir binary kütüphanesidir ve üzerinde değişiklik yapılamamaktadır. İçinde bulunan **`payInTurbo`** metodu kart numarası, tutar, ödenecek yer, taksit sayısı bilgilerini kullanarak ödeme işlemini yapmaktadır. Bu arayüzü kullanarak POS cihazları için ödeme işlemleri yapılmaktadır. Bunu için **`POSPayment`** sınıfı türetilmiştir.

`ModernPayment`: Yeni ödeme arayüzünü olarak karşımıza çıkmaktadır. Benzer şekilde kart numarası, tutar, ödenecek yer, taksit bilgilerini kullanarak ödeme işlemini yapacaktır. Bu arayüzü kullanarak online ödeme işlemleri yapılmaktadır. Bunun için **`OnlinePayment`** sınıfı türetilmiştir.

Fakat ödeme işlemleri için TurboPayment kullanılmak zorundadır. Hem yeni ödeme sistemine geçip hem de eskiyi kullanmak zorunda olduğumuzdan devreye tasarım örüntüleri girmektedir. Mevcut durumu kurtarmak için **Adapter Design Pattern** kullanılmıştır.

İki arayüz arasında uyarlayıcı bir rol oynayan **`PaymentAdapter`** tasarlanmıştır. Bu arayüz TurboPayment arayüzünü uygulamaktadır. Kurucu yöntem ile de ModernPayment objesi alır. İçindeki pay yöntemi ile de modernPayment tipindeki ödeme bilgilerini TurboPayment ödeme yöntemini kullanarak işlemi tamamlar.

UML Diagram



Ahmet Mert Gülbahçe
141044015

PART 2

Bu bölümde e-postalar için adres defteri tasarlanıp kod olarak uygulancaktır. Bu işlemler için **`Email`** isimli arayüz sınıfı oluşturulmuştur. Kişisel ve grupları içinde bulunduran e-posta listesi bu arayüzden türemektedir.

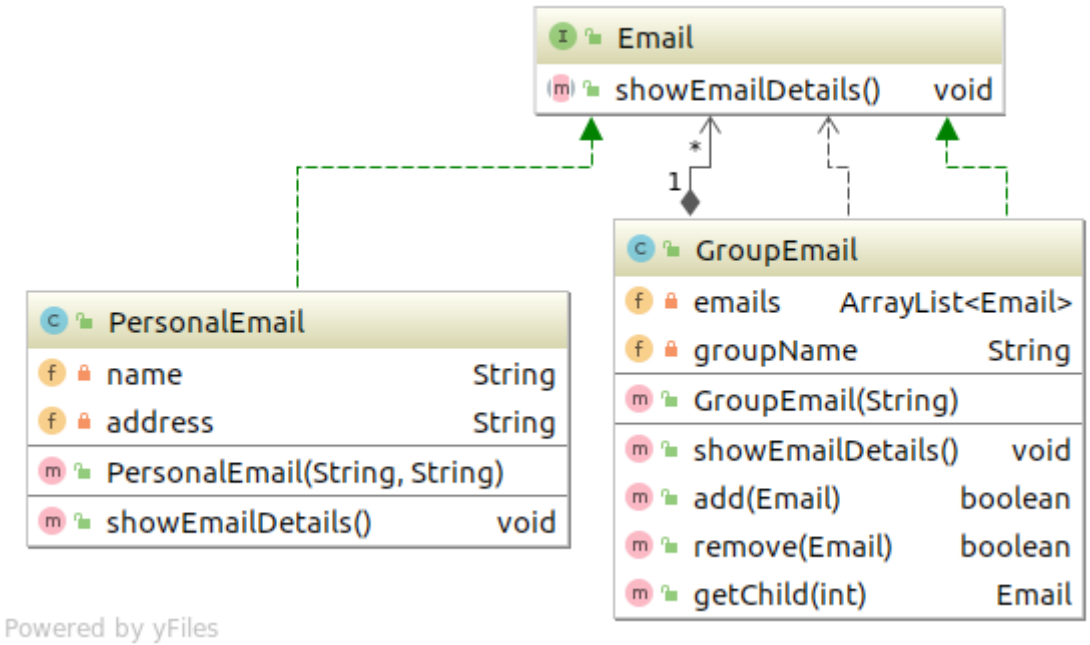
```
public interface Email {
     void showEmailDetails();
}
```

`PersonalEmail`: Veri olarak kişi adı ve e-posta adresini barındırmaktadır. Kurucu yöntem ile bu bilgilere atama yapılmaktadır. Arayüzden miras gelen `showEmailDetails()` yöntemi ile de bu bilgileri ekrana yazdırmaktadır.

`GroupEmail`: Veri olarak grup ismi ve e-postaları içinde tutmak için ArrayList bulunmaktadır. Kurucu yöntem ile grup ismi alır ve ArrayList'i kullanmak için hazır hale getirir. Ayrıca aşağıdaki gibi diğer yöntemleri vardır:

`boolean` `add(Email email)`: Grup içinde Email tipinde yeni elemanlar eklemektedir.
`boolean` `remove(Email email)`: Grup içindeki Email tipinde verilen elemanı silmektedir.
`Email getChild(`**`int`**` index)`: Verilen indisteki elemanı geri döndürmektedir.

UML Diagram



Ahmet Mert Gülbahçe
141044015

PART 3

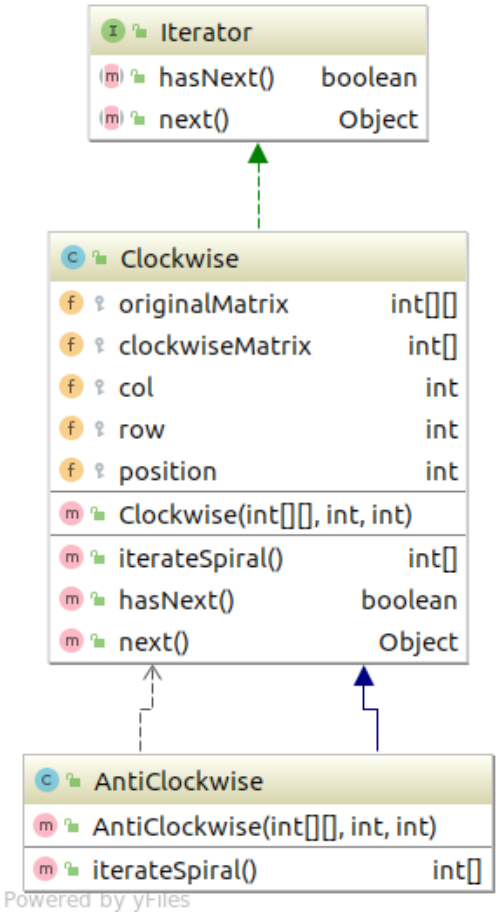
Bu bölümde, içinde int değerlerinin bulunduğu 2D array içinde saat yönünde ve saat yönünün tersinde iterasyon yapılması istenmiştir. İterasyon işlemi olduğundan **Iterator** tasarım örüntüsü kullanılmıştır.

Öncelikle Java'ya ait olmayan bir Iterator arayüzü tasarlandı. Ardından bu arayüzü gerçekleyen **`Clockwise`** ve **`AntiClockwise`** sınıfları türetildi. Bu sayede iki sınıf üzerinde iterasyon işlemlerini yapılması sağlandı.

`Clockwise`: Kurucu yöntem ile 2D int array almaktadır. Bu array üzerinde spiral olarak iterate işlemi yapılıp tek boyutlu bir array içinde saklanmaktadır. Array üzerinde gezinme işlemini yapan `iterateSpiral()` yönteminin kaynak kodu https://www.geeksforgeeks.org/print-a-given-matrix-in-spiral-form/ adresinden alınmış olup birkaç değişiklik uygulanmıştır.

`AntiClockwise`: Bu sınıf ise yukarıdaki Clockwise sınıfından türemiştir. Yaptıkları işler iterasyon işleminin tersten işlemesi haricinde aynıdır. O yüzden bu şekilde türemesi kod kalabalığından kurtarmış oldu. Yine aynı kaynak kod kullanılmış olup sadece tersten iterasyon işlemi için değiştirilmiştir.

UML Diagram



Ahmet Mert Gülbahçe
141044015

PART 4

Bu bölümde verilen matrisler için ayrık dönüşüm işlemlerini yapan sınıflar oluşturuldu. DFT ve DCT işlemlerinin yapılacağı bu bölümde, iki işlem de aynı işi yapıtğından ve aynı kategori altında bulunduğundan **Template** Tasarım Örüntüsü kullanıldı.

UML Diagram