ÖDEV 5

Murat, bir yazılım mühendisi olarak bir bankanın kredi kartı yönetim sistemini geliştirmekle görevlendirilmiştir. Banka, müşterilerine daha çekici hizmetler sunabilmek adına farklı türde kredi kartları tasarlamayı planlamaktadır. Bu kartlardan biri, müşterilerin yaptıkları harcamalardan puan kazanmalarını sağlayan bir "*Ödüllü Kredi Kartı*" olacaktır. Müşteriler, topladıkları bu puanları daha sonra alışverişlerde veya banka tarafından sunulan diğer avantajlar için kullanabilecektir.

Banka, bu projede kredi kartlarının temel özelliklerini kapsayan bir ebeveyn sınıf (*parent class*) olan **CreditCard** sınıfını halihazırda kullanmaktadır. Murat'tan bu sınıfa dayanarak yeni bir **RewardCreditCard** sınıfı (*yani bir alt sınıf (subclass)*) oluşturması istenmiştir.

Bilindiği üzere, **RewardCreditCard** sınıfı (*class*) **CreditCard** sınıfının customer, bank, acnt, limit, ve balance=0 örnek niteliklerini (*instance attributes*) kalıtım olarak alacaktır.

RewardCreditCard alt sınıfının iki yeni değişkeni gelecektir.

- **reward_rate**, müşterilerin harcamaları üzerinden belirli bir oranla puan kazanmalarını sağlayacaktır. Bu niteliğin varsayılan değeri şimdilik 0.01 olsun.
- **rewards_points**, toplam kazanılan puanları tutacaktır. DİKKAT, bu değişken sınıfın içerisinde kullanacağınız bir değişkendir (bu nedenle başına self öneki (*prefix*) almalıdır) ancak **RewardCreditCard** sınıfının *__init__* metoduna nitelik (*attribute*) olarak verilemez. Tabii ki, her müşterinin bu kartı kullanmaya başladığı anda ödül puanı 0 olmalıdır (yani, self.reward_points = 0)

Sizden üç ayrı örnek metodu (instance method) gerçekleştirmeniz beklenmektedir:

1. **get_rewards_points():** Bu metot bir müşterinin toplamda sahip olduğu ödül puanını döndürmelidir.
2. **redeem_rewards():** Bu metot, bir müşterinin hesabındaki ödül puanlar ile alışveriş yapmasını sağlamalıdır. Bu metot dışarıdan points isminde bir niteliği (attribute) kabul edecektir. Bu points değişkeni, alacağı ürünün puanını temsil etmektedir. Eğer müşterinin hesabında bulunan puanı, alacağı ürünün puanından az ise, ekrana "Yetersiz puan!" ibaresini döndürmelidir. Eğer müşterinin hesabındaki puan yeterli ise, alacağı ürünün puanı hesabındaki puandan düşmeli ve ekrana f"{points} puan başarıyla kullanıldı!" ibaresi döndürülmelidir.
3. **charge(price):** Hatırlayacağınız üzere ÖDEV 4'te CreditCard sınıfının charge isminde bir metodunu yazdınız. Burada ise RewardCreditCard sınıfının charge metodunu yazacaksınız. Burada da bu metot price isminde dışarıdan bir değişken alacaktır. Diğer bir deyişle, yapılan alışverişin ne kadar tuttuğu. Bu price değerini, sizin RewardCreditCard sınıfının bir niteliği olan reward_rate niteliği ile çarpmanız (price * self.reward_rate) ve bir alışverişten kazandığınız puanı bulmanız ve bu alışverişten kazanılan puanı, müşterinin mevcut puanlarına (rewards_points) eklemeniz gerekmektedir. Hatırlayınız, CreditCard sınıfının charge metodu f"İşlem Onaylandı!" string'ini döndürüyordu. Burada dikkat etmeniz nokta, eğer CreditCard sınıfının charge metodu f"İşlem Onaylandı!" olarak dönerse RewardCreditCard sınıfının charge metodunun çalışması gerektiğidir. RewardCreditCard sınıfının charge metodunun, alışveriş başarılı ise (yani işlem onaylanırsa) AYRI AYRI önce f"{result} {earned_points:.2f} puan kazandınız!" ibaresini ve daha sonra "İşlem Onaylandı!" ifadesini döndürmesidir. Eğer alışveriş başarısız ise, f"Limit yetersizdir!" ifadesini döndürmesi yeterlidir (çünkü müşterinin balance'ında yeterli para yoktur).

İKİNCİ SAYFAYA BAKINIZ.

Ekranda görmenizi istediğim çıktı şu şekildedir:

```
[4]: murat = RewardCreditCard(customer = 'Mustafa Murat ARAT', bank = 'Yapı Kredi Bankası', acnt = '5391 0375 9387 5309', limit = 100000, balance = 57000,  reward_rate=0.01)
murat
# <__main__.RewardCreditCard at 0x111230cd0>

murat._reward_rate
# 0.01

# Murat'ın başlangıçtaki toplam puanı
murat._rewards_points
# 0

# Murat 10,000 TL'lik ilk alışverişini gerçekleştiriyor ve bu alışverişinden 10000 * 0.01 = 100 puan kazanıyor.
murat.charge(price = 10000)
# 'İşlem Onaylandı! 100.00 puan kazandınız!'

# Murat'ın toplam puanı
murat._rewards_points
# 100

# Murat'ın hesabında kullanabileceği para
murat.get_balance()
# 47000

# Murat 30,000'lik ikinci alışverişini gerçekleştiriyor ve bu alışverişinden 30000 * 0.01 = 300 puan kazanıyor.
murat.charge(price = 30000)
# 'İşlem Onaylandı! 300.00 puan kazandınız!'

# Murat'ın toplam puanı
murat.get_rewards_points()
# 400.0

# Murat'ın hesabında kullanabileceği para
murat.get_balance()
# 17000

# Murat 20,000'lik üçüncü alışverişini gerçekleştiriyor. Ancak, hesabında sadece 17,000 TL para var
# Para yetmediği için 'Limit yetersizdir!' uyarısı alıyor.
murat.charge(price = 20000)
# 'Limit yetersizdir!'

# Murat 4,000 TL'lik kredi kartı ödemesi gerçekleştiriyor.
murat.make_payment(amount = 4000)
# 'Kredi kartı borç ödemesi için teşekkür ederiz.'

# Murat'ın hesabında kullanabileceği para
murat.get_balance()
# 21000

# Murat 20,000'lik üçüncü alışverişini bir kez daha gerçekleştiriyor. Artık yeterli parası var.
# Bu nedenle 'İşlem Onaylandı! 200.00 puan kazandınız!' ibaresini görebiliyor.
# ve bu alışverişinden 20000 * 0.01 = 200 puan kazanıyor.
murat.charge(price = 20000)
# 'İşlem Onaylandı! 200.00 puan kazandınız!'

# Murat'ın hesabında kullanabileceği para
murat.get_balance()
# 1000

# Murat'ın toplam puanı
murat.get_rewards_points()
# 600.0
```

```
[4]: 600.0
```