NESNEYE YÖNELİK PROGRAMLAMA LAB. 6

Restoran Sipariş ve Menü Yönetim Sistemi

1. **Senaryo**

Bir restoran için **menü ve sipariş yönetim sistemi** tasarlıyorsun. Restoranda:

- Farklı türde ürünler var: **Yemek**, **İçecek**, **Tatlı**.
- Bazı ürünler **indirimli** olabilir.
- Bazı ürünler **hazırlanması daha uzun** olan "özel ürünler".
- Tüm ürünler için bazı ortak bilgiler var (isim, fiyat, hazırlama süresi vs.), ama hesaplama mantıkları ve bazı davranışlar ürün türüne göre değişiyor.

2. Temel Yapılar

2.1. final class Sabitler

Bu sınıf miras alınamayacak (final). İçinde sadece sabitler olsun:

```
public final class Sabitler {
   public static final double KDV_ORANI = 0.10; // %10
   public static final double SERVIS_UCRETI_ORANI = 0.05; // %5
   public static final int MIN_HAZIRLAMA_SURESI = 5; // dakika
   private Sabitler() {
      // Nesne üretilmesin diye private constructor
   }
}
```

2.2. abstract class Urun

Tüm menü ürünlerinin ortak atası.

Alanlar:

- protected String ad;
- protected double hamFiyat; // KDV ve servis ücreti hariç
- protected int hazirlamaSuresi; // dakika

Kurucular (constructor):

- Tüm alanları alan bir kurucu (ad, hamFiyat, hazirlamaSuresi).
    - Eğer hazirlamaSuresi, Sabitler.MIN_HAZIRLAMA_SURESI'nden küçük verilirse, alanı minimum değere eşitle (ör: this.hazirlamaSuresi = Sabitler.MIN_HAZIRLAMA_SURESI;).

Metotlar:

- Getter'lar:
    - getAd(), getHamFiyat(), getHazirlamaSuresi()
- public void setHamFiyat(double hamFiyat)
    - Fiyat negatif verilirse, 0'a ayarla.
- public void urunBilgisiYazdir()
    - ürün adı, ham fiyat, hazırlama süresi yazdırılır.
- public abstract double satisFiyatiHesapla();
    - Her alt sınıfın, KDV + servis vs. dahil **kendi satış fiyatını** hesaplaması isteniyor.
- public abstract String kategoriAdi();
    - Örn: "Yemek", "İçecek", "Tatlı".

2.3. Arayüzler

2.3.1. interface IndirimUygulanabilir

```
public interface IndirimUygulanabilir {
   void indirimUygula(double oran); // oran 0.0 - 1.0 arası, örn 0.2 = %20 indirim
}
```

Bu arayüzü implemente eden ürünler, kendi hamFiyat değeri üzerinde indirimi uygular.

2.3.2. interface OzelUrun

```
public interface OzelUrun {
   String ozelNotAl();       // Müşteriye gösterilecek özel not (örn: "Şefin önerisi")
   void ozelHazirlikYap();   // Konsolda özel hazırlık adımlarını yazdır.
}
```

Özel menü ürünleri için kullanılacak.

3. Alt Sınıflar

3.1. class Yemek extends Urun implements IndirimUygulanabilir, OzelUrun

Ek Alanlar:

- private boolean vegMi; // vejetaryen mi
- private boolean ozelGunYemegiMi; // örn: günün menüsü

Kurucu:

- Tüm alanları alan bir kurucu yaz.
- ozelGunYemegiMi == true ise, hamFiyat %15 daha düşük başlatılabilir (opsiyonel mantık).
-

Metotlar:

- @Override public double satisFiyatiHesapla() Öneri:
    - double fiyat = hamFiyat;
    - Üzerine KDV ekle: fiyat += fiyat * Sabitler.KDV_ORANI;
    - Üzerine servis ücreti ekle: fiyat += fiyat * Sabitler.SERVIS_UCRETI_ORANI;
    - Eğer vegMi == true ise, fiyatı +2 TL artır (özel malzemeler var varsayalım).
- @Override public String kategoriAdi() → "Yemek"
- @Override public void indirimUygula(double oran)
    - hamFiyat = hamFiyat * (1 - oran);
    - Oran 0–1 dışıysa, hiç uygulama (kontrol koy).
- @Override public String ozelNotAl()
    - Örn: "Şefin özel sosuyla hazırlanır." veya ozelGunYemegiMi durumuna göre farklı metin.
- @Override public void ozelHazirlikYap()
    - Konsola birkaç satır hazırlık adımı yaz (örn: "Yemek için özel marine hazırlanıyor...").

3.2. class Icecek extends Urun implements IndirimUygulanabilir

Ek Alanlar:

- private boolean sogukMu;
- private boolean sekerliMi;
- private static int toplamIcecekSayisi = 0;
    - Üretilen tüm Icecek nesnelerini saymak için.

Kurucu:

- Tüm alanları alan kurucu.
- Her oluşturulduğunda toplamIcecekSayisi'ni 1 artır.

Metotlar:

- @Override public double satisFiyatiHesapla() Örnek mantık:
    - Temel: fiyat = hamFiyat + hamFiyat * Sabitler.KDV_ORANI;
    - Sıcak içecekler için (sogukMu == false) ekstra 1 TL ekle (sunum farkı).
    - Şekersiz içeceklerde (sekerliMi == false) 0.5 TL indirim yap.
- @Override public String kategoriAdi() → "İçecek"
- @Override public void indirimUygula(double oran)

    - Benzer şekilde hamFiyat üzerinde çalışır.
- public static int getToplamIcecekSayisi()
    - Şu ana kadar üretilen içecek sayısını döndürür.

3.3. class Tatli extends Urun implements OzelUrun

Ek Alanlar:

- private boolean sutluMu;
- private boolean sicakServisMi;

Metotlar:

- @Override public double satisFiyatiHesapla()

Öneri:

    - fiyat = hamFiyat + hamFiyat * Sabitler.KDV_ORANI;
    - Sütlü tatlılarda (sutluMu == true) maliyet yüksek olsun, +2 TL ekle.
    - Sıcak servis ise (sicakServisMi == true) +1 TL ekle.
- @Override public String kategoriAdi() → "Tatlı"
- @Override public String ozelNotAl()
    - Örn: "Sıcak servis edilir." veya alanlara göre dinamik bir not.
- @Override public void ozelHazirlikYap()
    - Konsola özel hazırlık adımlarını yazdır.

4. class Menu

Restoran menüsünü tutan sınıf.

Alanlar:

- private String ad; // Menü adı (örn: "Akşam Menüsü")
- private ArrayList<Yemek> yemekler;
- private ArrayList<Icecek> icecekler;
- private ArrayList<Tatli> tatlilar;

Kurucu:

- Menü adını alan bir kurucu.
- Listeleri başlangıçta oluştur.

Metotlar:

- public void yemekEkle(Yemek yemek)
- public void icecekEkle(Icecek icecek)

- public void tatliEkle(Tatli tatli)
- public void menuYazdir()
    - Tüm yemekleri, içecekleri, tatlıları kategori başlıklarıyla birlikte ekrana yazdır.
    - Her ürün için:
        - urunBilgisiYazdir()
        - System.out.println("Kategori: " + kategoriAdi());
        - System.out.println("Satış Fiyatı: " + satisFiyatiHesapla());
- public double toplamMenuFiyati()
    - Listedeki tüm ürünlerin satış fiyatlarını toplayıp döndür.

Burada dikkat: Her liste kendi alt sınıf tipini tutuyor. Üst sınıf referansı (Urun) ile tek bir liste yapmıyoruz.

5. class Siparis

Bir müşteri siparişi.

Alanlar:

- private String musteriAdi;
- private ArrayList<Yemek> siparisYemekler;
- private ArrayList<Icecek> siparisIcecekler;
- private ArrayList<Tatli> siparisTatlilar;
- private boolean paketServisMi;

Kurucu:

- Müşteri adını ve paket servis bilgisini alan kurucu.

Metotlar:

- Ekleme metotları:
    - public void yemekEkle(Yemek yemek)
    - public void icecekEkle(Icecek icecek)
    - public void tatliEkle(Tatli tatli)
- public double toplamTutarHesapla()
    - Tüm ürünlerin satış fiyatlarını toplar.
    - Eğer paketServisMi == true ise, toplamın üzerine sabit 10 TL paket ücreti ekler.
- public void siparisOzetiYazdir()
    - Müşteri adı
    - Paket mi salon mu

- Her kategori için ürün listesi + fiyatı
- Son toplam tutar

6. Main Sınıfı (public class Uygulama)

main metodunda:

1. Bir Menu oluştur, içine:
    - En az 2 Yemek
    - En az 2 Icecek
    - En az 2 Tatli ekle.
2. Bazı ürünler için:
    - indirimUygula çağır (sadece Yemek ve Icecek için geçerli).
    - ozelHazirlikYap ve ozelNotAl metotlarını çağır (Yemek ve Tatli için).
3. Menüyü yazdır:
    - menu.menuYazdir();
    - Menünün toplam fiyatını ekrana yaz (menu.toplamMenuFiyati()).
4. Birkaç ürün seçerek bir Siparis oluştur:
    - Siparis içinde yemekEkle, icecekEkle, tatliEkle metotlarını kullan.
    - siparisOzetiYazdir() ile siparişi ekrana yazdır.
5. Program sonunda toplam içecek sayısını yaz:

System.out.println("Şu ana kadar üretilen Icecek sayısı: " + Icecek.getToplamIcecekSayisi());