



BURSA TEKNİK ÜNİVERSİTESİ

Melike Şevval Şinar

23360859039

Şube 2

Restoran Simülasyonu

1. Classları Oluşturma

Projemde ilk olarak UML diyagramlarına uygun olarak classlarımı oluşturmakla başladım.

```
melike.java Siparis.java Yemek.java Icecek.java *Restoran.java × Muste
1 package deneme;
2 import java.util.ArrayList;
3
4
5
6 public class Restoran {
7     private ArrayList<Urun> menu= new ArrayList<Urun>();
8     private ArrayList<Siparis> siparisler= new ArrayList<Siparis>();
9     private ArrayList<Musteri> musteriler= new ArrayList<Musteri>();
10    private ArrayList<Garson> garsonlar= new ArrayList<Garson>();
11
12    Restoran(){}
13    public void menuyeUrunEkle(Urun urun) {
14        menu.add(urun);
15        menu.toArray();
16    }
17    public void müşteriEkle(Musteri musteri) {
18        musteriler.add(musteri);
19    }
20    public void garsonEkle(Garson garson) {
21        garsonlar.add(garson);
22    }
23    public void menuyuGoster() {
24        System.out.println("---MENU---");
25        for(Urun yemek :menu) {
26            yemek.urunBilgisi();
27        }
28    }
29 }
```

Resim 1.1 Restoran Classı

```
melike.java × Siparis.java Yemek.java Icecek.java × Restoran.java Musteri.java Urun.java Garson.j
deneme/src/deneme/melike.java
2
3 public class Icecek extends Urun {
4     private String boy;
5
6     Icecek (String ad,double fiyat,String boy){
7         super(ad,fiyat);
8         this.boy=boy;
9     }
10    public void urunBilgisi() {
11        System.out.println("İçecek: "+ ad+"\nFiyat: "+fiyat +"\nBoyut: "+ boy+"\n- - - - -");
12    }
13
14 }
15 }
```

Resim 1.2 Icecek Classı

```

1 package deneme;
2 import java.io.*;
3 import java.io.File;
4 import java.io.FileWriter;
5 import java.io.PrintWriter;
6
7 public class Garson extends Kisi {
8     Garson( String ad){
9         super(ad);
10    }
11    public void siparisAl(Siparis s,Siparis siparisler) {

```

Resim 1.3 Garson Classı

```

melike.java Siparis.java x Yemek.java Icecek.java Restoran.java Musteri.java Urn
1 package deneme;
2 import java.util.Date;
3
4 public class Siparis {
5
6     private Yemek yemek;
7     private Icecek icecek;
8     public Garson garson;
9     private Musteri musteri;
10    public Date tarih;
11    public static long siparis_no;
12    Date saat=new Date();
13    Siparis(){ }
14    Siparis (Yemek yemek,Icecek icecek, Garson garson,Musteri musteri,Date tarih) {
15        this.yemek=yemek;
16        this.icecek=icecek;
17        this.garson=garson;
18        this.musteri=musteri;
19        this.tarih=tarih;
20    }

```

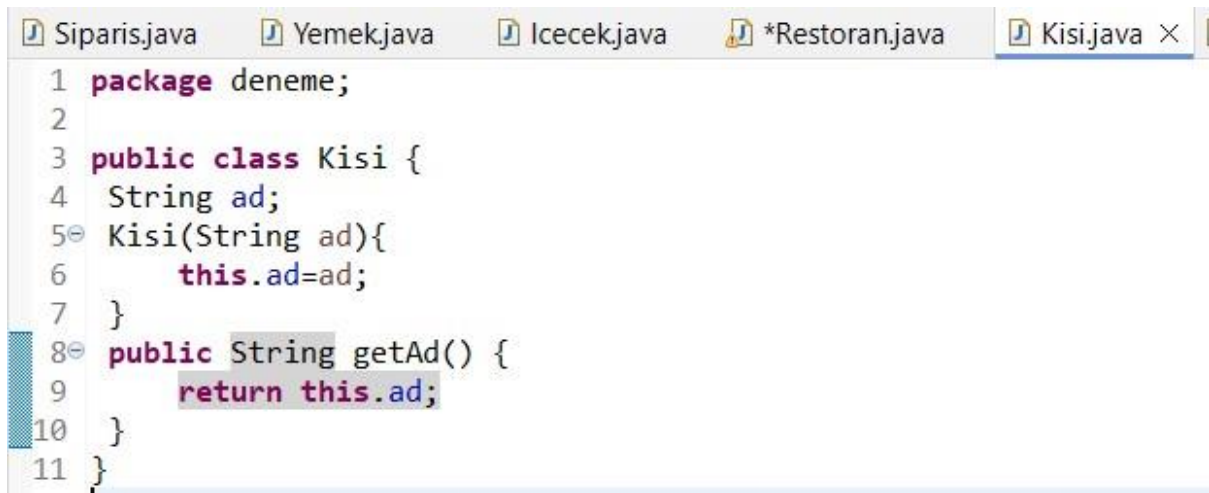
Resim 1.4 Siparis Classı

```

melike.java Siparis.java Yemek.java x Icecek.java Restoran.java Musteri.java Urun.java
1 package deneme;
2
3 public class Yemek extends Urun {
4     private String tur;
5     Yemek(String ad,double fiyat,String tur){
6         super(ad,fiyat);
7         this.tur=tur;
8     }
9     public void urunBilgisi() {
10        System.out.println("Yemek: "+ ad+"\nFiyat: "+fiyat +"\nTür: "+ tur+"\n- - - -");
11    }
12 }
13 }

```

Resim 1.5 Yemek Classı




```

1 package deneme;
2
3 public class Kisi {
4     String ad;
5     Kisi(String ad){
6         this.ad=ad;
7     }
8     public String getAd() {
9         return this.ad;
10    }
11 }

```

Resim1.6 Kisi Classı

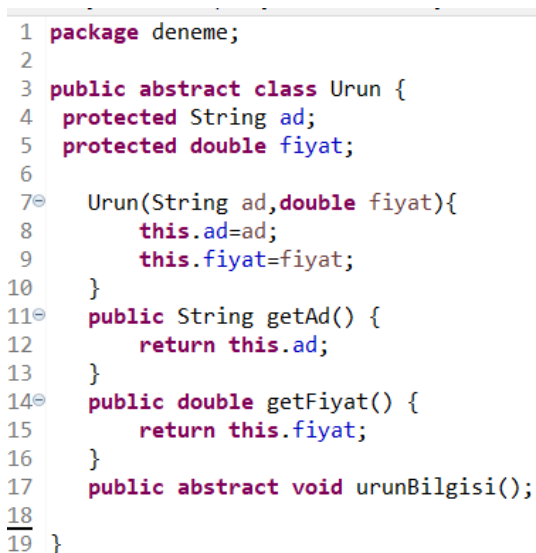


```

1 package deneme;
2
3 import java.util.Date;
4
5 public class Musteri extends Kisi {
6     Musteri(String ad){
7         super(ad);
8     }
9     public Siparis siparisVer(Yemek yemek, Icecek icecek) {
10         return new Siparis(yemek,icecek, null, this, new Date());
11     }
12 }

```

Resim 1.7 Musteri Classı



```

1 package deneme;
2
3 public abstract class Urun {
4     protected String ad;
5     protected double fiyat;
6
7     Urun(String ad,double fiyat){
8         this.ad=ad;
9         this.fiyat=fiyat;
10    }
11    public String getAd() {
12        return this.ad;
13    }
14    public double getFiyat() {
15        return this.fiyat;
16    }
17    public abstract void urunBilgisi();
18
19 }

```

Resim 1.8 Urun Classı

Bu şekilde Siparis, Restoran, Urun, Yemek, Icecek, Kisi, Musteri ve Garson olmak üzere 8 tane class oluşturdum ve bunların UML diyagramındaki değişkenlerini ekledim. Bu classlardan Yemek ve Icecek Urun classının mirasçısı; Garson ve Musteri ise Kisi classının mirasçısı olacak şekilde yaptım.

Değişkenleri tanımladıktan sonra ise bu classlara ait constructorları tanımladım. Bu classlara ait gerekli metotları ekledim.

2. Menü Oluşturma ve Yazdırma

Gerekli Classları tanımladıktan sonra menüyü oluşturup yazdırmak için ilk olarak test classımda Yemek ve Icecek Classlarından nesneler oluşturdum. Bu nesnelerde menüye ekleyeceğimiz yemek ve içecekler yazıyor. Daha sonrasında da bir Restoran nesnesi oluşturdum. Restoran classımdaki menuyeUrunEkle isimli metotumu kullanmak amacıyla bu restoran nesnesini oluşturdum. Bu metot sayesinde menüye yemek ve içecek ekleyebiliyoruz.

```
public static void main (String[] args) {  
    Urun y1= new Yemek("Margarita",110,"pizza");  
    Urun y2= new Yemek("Pepperoni",130,"pizza");  
    Urun y3= new Yemek("BigMac",120,"hamburger");  
    Urun i1= new Icecek("Kola",15,"kutu");  
    Urun i2= new Icecek("Kola",20,"şişe");  
    Urun i3= new Icecek("Ayran",10,"kucuk");  
    Urun i4= new Icecek("Ayran",15,"buyuk");  
    Musteri m1=new Musteri("Ayşe Hanım");  
    Musteri m2=new Musteri("Selin Hanım");  
    Musteri m3=new Musteri("Kaan Bey");  
    Garson g1=new Garson("Serkan Bey");  
    Garson g2=new Garson("Nazlı Hanım");  
    Garson g3=new Garson("Ahmet Bey");  
  
    Restoran r1= new Restoran();
```

Resim 2.1 Test Sınıfında Nesne Oluşturma

```
Restoran(){  
    public void menuyeUrunEkle(Urun urun) {  
        menu.add(urun);  
        menu.toArray();  
    }  
}
```

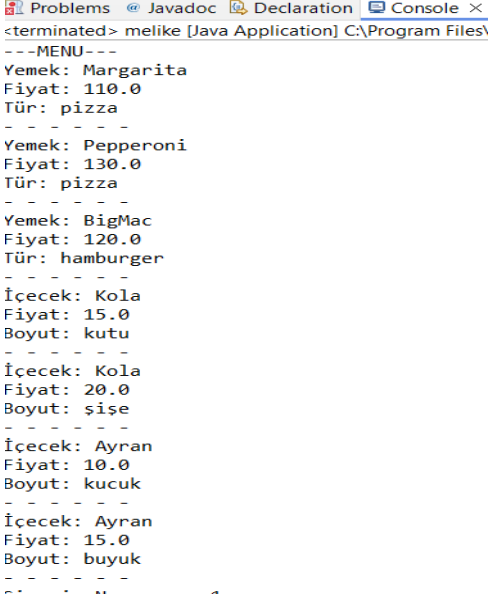
Resim 2.2 Menüye Ürün Ekle Metodu

Ürünlerimizi menüye ekledikten sonra Restoran classının içindeki menuyuGoster metodu menüyü konsola yazdırıyor. Bu metotta urunBilgisi adında başka bir metottan da yararlandım. Bu metod bize yemeğin türünü ve içeceğin boyutunu belirtiyor. Bu şekilde menümü konsola yazdırmış oldum.

```

public void menuyuGoster() {
    System.out.println("---MENU---");
    for(Urun yemek :menu) {
        yemek.urunBilgisi();
    }
}

```



Resim 2.3 Menü Konsol Çıktısı

3.Rastgele Siparişler Oluşturma ve Yazdırma

Siparişlerimizde sipariş verilen ürünlerin yanı sıra garson ve müşteri bilgisi de bulunması gerekiyor. Bu sebeple Garson ve Müşteri nesneleri de oluşturdum. Yine Restoran classının içinde bulunan müşteriler ve garsonlar isimli arraylistlere garsonEkle ve musteriekle metodlarından yararlanarak 3 adet garson ve 3 adet müşteri ekledim.

Restoran Classının içindeki rastgeleSiparisOlustur metodunda ise yemekler, içecekler, garsonlar, müşteriler arraylistlerinden random aldıklarımızla siparişlerimi oluşturuyorum.

```

public void rastgeleSiparisOlustur() {
    Random random = new Random();

    for (int i = 0; i < 10; i++) {

        int siparisTuru = random.nextInt(3); // 0: sadece yemek, 1: sadece içecek, 2: yemek ve içecek
        Yemek ryemek = null;
        Icecek ricecek = null;

        if (siparisTuru == 0) {
            ryemek= (Yemek) menu.get(random.nextInt(3));
        } else if (siparisTuru == 1) {
            ricecek= (Icecek) menu.get(random.nextInt(3) + 3);
        } else if (siparisTuru == 2) {
            ryemek= (Yemek) menu.get(random.nextInt(3));
            ricecek= (Icecek) menu.get(random.nextInt(3) + 3);
        }

        int rmusteri=random.nextInt(musteriler.size());
        int rgarson=random.nextInt(garsonlar.size());
        Musteri musteris = musteriler.get(rmusteri);

        Garson garson = garsonlar.get(rgarson);
        Siparis siparis = musteris.siparisVer(ryemek, ricecek);
        siparis.garson = garson;
        siparis.siparis_no = i+1;
        garson.siparisAl(siparis, siparis);
    }
}

```

Resim 3.1 Rastgele Sipariş Oluştur Metodu

Bu metotta ilk başta sadece yemek sadece içecek veya hem yemek hem içecek seçimini oluşturmak için 0, 1, 2 sayılarından birini random seçtiriyorum. Eğer 0 çıkarsa sadece yemek 1 çıkarsa sadece içecek 2 çıkarsa hem yemek hem içecek seçimi yapılmış oluyor. Sonrasında müşteriler ve garsonlar arraylistlerinden rastgele garson ve müşteri seçimi yapıyor.

Garson Classındaki siparisAl metotunu da burada kullandım bu metot rastgele siparişlerimizi sipariş numarasına göre ayrı ayrı dosyalara yazdırıyor. Yani test classımızda rastgeleSiparislerOlusturu kullandığımızda içindeki siparisAl metoduyla da beraber siparişleri oluşturup ayrı ayrı dosyalara da yazdırmış oluyoruz.

```
public void siparisAl(Siparis s,Siparis siparisler) {
    s.garson= this;
    s.siparisBilgisi();

    String klasorAdi = "Siparisler";
    File klasor = new File(klasorAdi);
    if (!klasor.exists()) {
        klasor.mkdir();
    }

    String dosyaAdi = klasorAdi + "/siparis_" + s.siparis_no + ".txt";
    try(    PrintWriter yazici = new PrintWriter(new FileWriter(dosyaAdi))){
        s.siparisYazdir(yazici);

    } catch (IOException e) {
    }
}
```

Resim 3.2 Sipariş Al Metodu

siparisAl metodumda önce Siparisler isimli bir klasör oluşturdum. Bu klasörün içinde oluşturduğumuz 10 tane rastgele siparişler olacak. Siparis classındaki siparisYazdir metodunu kullandım. Bu şekilde dosyalara sipariş bilgilerini yazdırdım.

```
public void siparisYazdir(PrintWriter yazici) {

    yazici.println("Sipariş Numarası: " + siparis_no);
    yazici.println("Müşteri: " + muster_i.getAd());
    yazici.println("Garson: " + garson.getAd());
    yazici.println("Yemek: " + (yemek != null ? yemek.getAd() : "-"));
    yazici.println("İçecek: " + (icecek != null ?  ıcecek.getAd() : "-"));
    yazici.println("Tarih: " + tarih);
}
```

Resim 3.3 Sipariş Yazdır Metodu

Siparişlerimi konsola ise yine Siparis classında olan siparisBilgisi metodu yardımıyla yazdırıyorum.

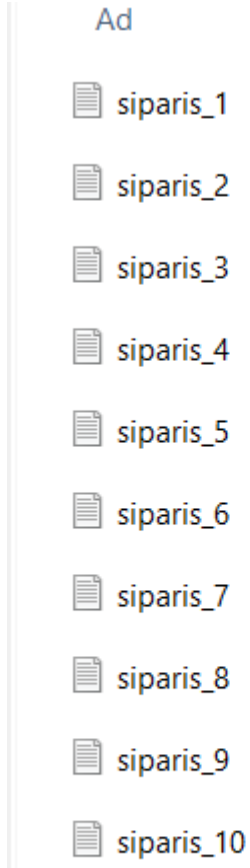
```
public void siparisBilgisi() {  
    System.out.println("Sipariş Numarası: " + siparis_no );  
    System.out.println("Müşteri: " + musteri.getAd());  
    System.out.println("Garson: " + garson.getAd());  
    System.out.println("Yemek: " + (yemek != null ? yemek.getAd() : "-"));  
    System.out.println("İçecek: " + (icecek != null ? icedek.getAd() : "-"));  
    System.out.println("Tarih: " + tarih);  
    System.out.println("-----");  
}
```

Resim 3.4 Sipariş Bilgisi Metodu

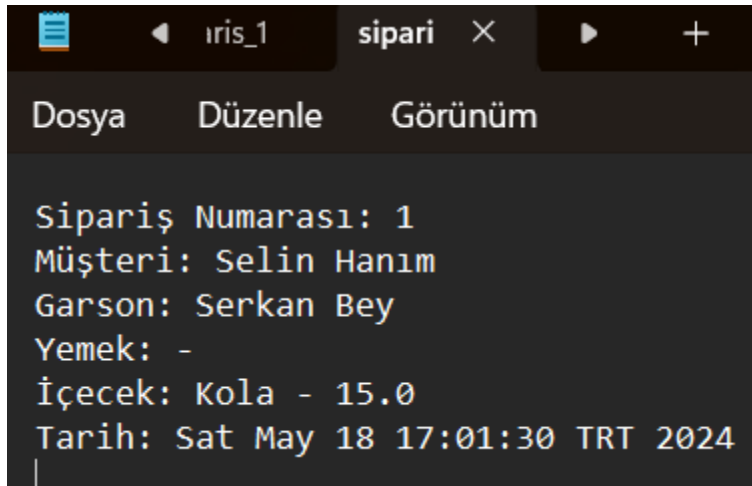


```
Problems @ Javadoc Declaration Console X  
<terminated> melike [Java Application] C:\Program Files  
  
Sipariş Numarası: 1  
Müşteri: Selin Hanım  
Garson: Serkan Bey  
Yemek: -  
İçecek: Kola - 15.0  
Tarih: Sat May 18 17:01:30 TRT 2024  
-----  
Sipariş Numarası: 2  
Müşteri: Kaan Bey  
Garson: Nazlı Hanım  
Yemek: -  
İçecek: Ayran - 10.0  
Tarih: Sat May 18 17:01:30 TRT 2024  
-----  
Sipariş Numarası: 3  
Müşteri: Kaan Bey  
Garson: Serkan Bey  
Yemek: Pepperoni - 130.0  
İçecek: -  
Tarih: Sat May 18 17:01:30 TRT 2024  
-----  
Sipariş Numarası: 4  
Müşteri: Kaan Bey  
Garson: Ahmet Bey  
Yemek: BigMac - 120.0  
İçecek: -  
Tarih: Sat May 18 17:01:30 TRT 2024  
-----  
Sipariş Numarası: 5  
Müşteri: Selin Hanım  
Garson: Ahmet Bey  
Yemek: BigMac - 120.0  
İçecek: -  
Tarih: Sat May 18 17:01:30 TRT 2024  
-----  
Sipariş Numarası: 6  
Müşteri: Ayşe Hanım  
Garson: Ahmet Bey  
Yemek: -  
İçecek: Kola - 15.0  
Tarih: Sat May 18 17:01:30 TRT 2024  
-----  
Sipariş Numarası: 7  
Müşteri: Ayşe Hanım  
Garson: Ahmet Bey  
Yemek: BigMac - 120.0
```

Resim 3.5 Siparişler Konsol Çıktısı



Resim 3.6 Siparişler Klasöründe Oluşmuş Sipariş Dosyaları



Resim 3.7 Dosyaya Yazılmış Siparişlerden Biri

4. Kaynakça

<https://drive.google.com/file/d/1K2O148G9tmqqCAz9XyHG4Y2xH9AM2rKd/view>

Daniel Liang Introduction to Java Programming