KOCAELİ ÜNİVERSİTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ
YAZILIM LABORATUVARI

Tuğba ÇEVİKER
210202056

Samethan KAZANBAŞ
210202043

I. ÖZET

Bu döküman Yazılım Laboratuvarının üçüncü projesi olan Restoran Yönetim Sisteminin çözümünü açıklamaya yönelik oluşturulmuştur. Dökümanda özet , katkılar , yöntem ve kaynakça kısımlarına yer verilmiştir.

II. YÖNTEM

Projede öncelikle bizden problem 1 ve problem 2 için ve bunlara ulaşmak için 2 butonlu bir arayüz tasarladık. Bu butonlar ilgili problemin çözümü için ilgili arayüze yönlendirme yapmamızı sağlıyor.

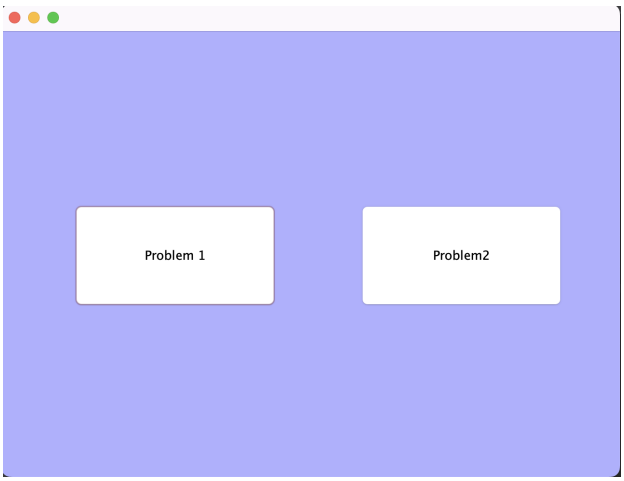


Fig. 1. Projemizin Anapaneli

PROBLEM 1

İlk olarak bu arayüzde müşteri ekle geri ve başlat butonları yer alıyor. Müşteri ekle butonu sayesinde Problem 1 için simülasyonumuza ilgili müşterileri ekleyebiliyoruz. Yazdığımız algoritmada her bir müşteri için müşterinin id si, adı ve öncelik durumu(true ya da false) kullanıcıdan isteniyor. Girilen değerlere göre müşteriler ekranda yazdırılıyor ve kullanıcının maksimum 10 adet müşteri girmesini gerçekleştirdiğimiz koşul ile bu kısım çalışıyor. Ayrıca bu arayüzün altında oluşturduğumuz classlarda yer alan bekleme süreleri yer alıyor. Kullanıcı dilerse bu değerleri değiştirebiliyor. Kullanıcı istediği sayıda müşteri eklediği ve süreleri de istediği şekilde değiştirdiğinde başlat butonuna basarak simülasyonu başlatır.

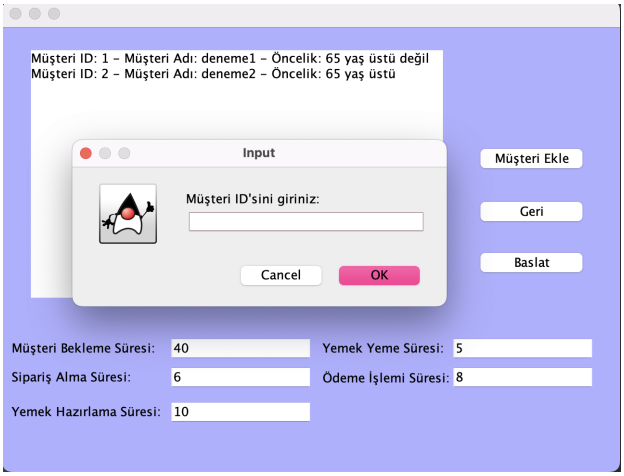


Fig. 2. Kullanıcıları Aldığımız ve Özel Durumlarını Kontrol Ettiğimiz Arayüz

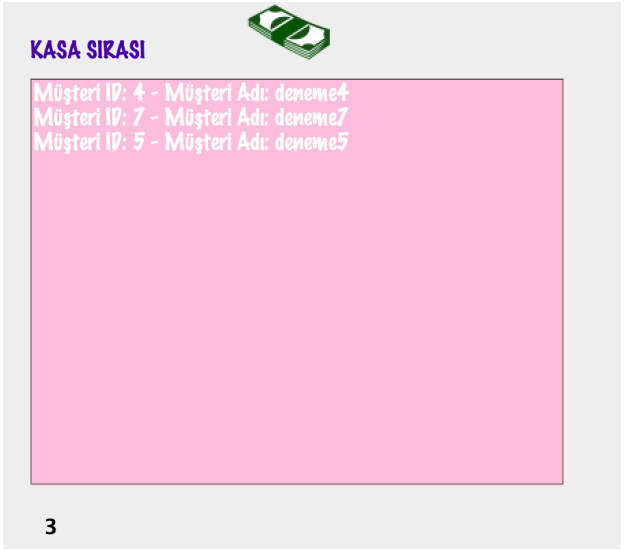
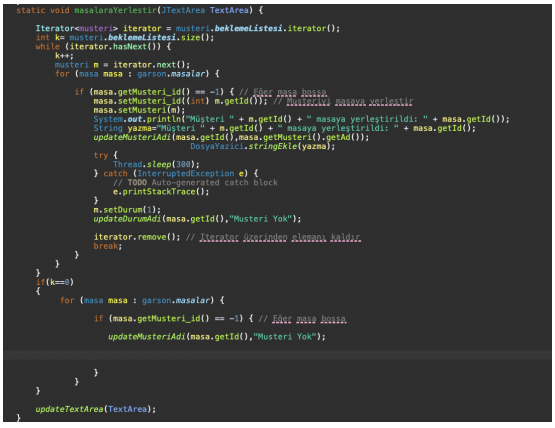


Fig. 3. Kasa Sırası ve Ödeme Yapan Müşterilerin Sayacı

Main THREAD

Başlat butonuna basıldığında ilk olarak önceki ekranda eklediğimiz müşteriler bir arraylist oarak yeni classımıza gönderilir. Bu sınıfta bu arrraylist belli operasyonlarla id, ad, ve öncelik durumu değerlerine ayrılarak müşterilerin her biri musteriler türünden değer alan arrayliste teker teker eklenir. 6 tane masa da oluşturulup masalar arraylistine eklenir. 3 Garson oluşturulup garsonlar arraylistine eklenir. 2 aşçı oluşturulup aşçılar arraylistine eklenir. kasa sınıfından da bir nesne oluşturulur. Ardından garson,aşçı,musteri ve kasa sınıfına ait tüm threadler .start() methoduyla başlatılır. Bekleme listesindeki her müşteri için bir timer oluşturulur ve oluşturulan bu timerla önceden belirlenen müşteri bekleme süresi kontrlü yapılır. eğer müşteri hala masaya oturmamışsa bu müşteri beklemelistesinden de kaldırılır ve ilgili thread yok edilir.masalarayerlestir methoduyla da bekleme listesinin en üst kısımdaki kişiler sırayla gezilerek boş masalarla eşleştirilir ve atamaları yapılır.

Fig. 4. Müşteri Bekleme Listesi



```
static void masalaraYerlestir(JTextArea TextArea) {

    Iterator<musteri> iterator = musteri.beklemeListesi.iterator();
    int k= musteri.beklemeListesi.size();
    while (iterator.hasNext()) {
        k++;
        musteri m = iterator.next();
        for (masa masa : garson.masalar) {

            if (masa.getMusteri_id() == -1) { // Eğer masa boşsa
                masa.setMusteri_id((int) m.getId()); // Müşteriyi masaya yerleştir
                masa.setMusteri(m);
                System.out.println("Müşteri " + m.getId() + " masaya yerleştirildi: " + masa.getId());
                String yazma="Müşteri " + m.getId() + " masaya yerleştirildi: " + masa.getId();
                updateMusteriAdi(masa.getId(),masa.getMusteri().getAd());
                                    DosyaYazici.stringEkle(yazma);

                try {
                    Thread.sleep(300);
                } catch (InterruptedException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
                m.setDurum(1);
                updateDurumAdi(masa.getId(),"Musteri Yok");

                iterator.remove(); // Iterator üzerinden elemanı kaldır
                break;
            }
        }
    }
    if(k==0)
    {
        for (masa masa : garson.masalar) {

            if (masa.getMusteri_id() == -1) { // Eğer masa boşsa

               updateMusteriAdi(masa.getId(),"Musteri Yok");


            }
        }
    }
    updateTextArea(TextArea);
}
```

Fig. 5. Masalara Yerleştir Methodu

Müşteri THREADLERİ

Müşteri Threadinin kurucu metodunda ilgili müşterinin önceliğine göre eğer önceliği varsa belemelistesinin başına eklenir. Yoksa bu listenin sonuna eklenir. Böylece öncelik durumuna göre müşteriler masalara yerleştirilecektir. Bu threadde ayrıca müşterilerin durumu kontrol edilerek durumları ekrana yazdırıır. Yemek yyor durumuna gelen müşteri bu kısımda belirlenen süre boyunca bekletilir sonrasında kasas sırasında bekleme durumuna geçer. Bu kısımda da ilgili müşteri kasabeklemesırası arraylistine eklenir.

Garson THREADLERİ

Garsonlar anık olarak masalar arrylistini gezer. Senkronize olması için masalar senkronize edilir. Masalardaki müşteriler gezilir ve eğer müşteri beklemedeyse garson ilk bulduğu müşteriyle eşleşir ve bu müşterinin durumunu günceller. Kendisini bu müşterinin gaarsonu olarak atar. Belirlenen sipariş alma süresi geçtikten sonra ilgili garson aşçılar arraylistini gezerek ilgili siparişi aşçıya iletmeye çalışır. Eğer iki aşçı da doluysa garson threadi aşçının boş duruma geçmesi beklenir. İletim olduğunda garson ilgili müşterinin siparişi hazırlamasını bekler. Sipariş hazır olunca garson yemeği ilgili müşteriye iletir ve durumunu günceller.

Aşçı THREADLERİ

Aşçılar bekleme halindedir. Garsonlar kendilerine sipariş getirdiğinde başta belirlenen sürede yemeği hazırlar ve ilgili müşterinin durumunu günceller. Bu güncellemenin ardından aşçı tekrar boş duruma düşer. Her aşçı maks 2 adet yemek yapar. Aşçılar doluysa garsonlar aşçıları bekler.

Kasa THREAD

Kasa her zaman kasabeklemelistesini kontrol eder eğer bu kısma bir müşteri eklendiyse müşteriyi işleme alır. Başta belirlenen süre sonunda kasa müşterinin ödemesini alır ve bu müşteriyi kasabeklemelistesinden kaldırır. AArdından bu müşterinin threadini yok eder. İlgili müşteri restorandan ayrılmış olur.Ardından müşterinin masası da boşaltılır. Ardından kasa masalara yerleştir methodunu tekrardan çağırarak ilgili masaya eğer bekleme listesinde biri varsa yerleştirmesini sağlar. Garsonlar tekrardan boş masaları gezer ve aşçılara siparişlerini iletir.

ARAYÜZ

Arayüzde threadlerden alınan verileri dinamikleştirmek için her labelı public olarak tanımladık. Bu labellarda yazan metinler de public olarak yazıldı. updatelabel methodlarıyla istediğimiz threaddten labelımızı güncelleyebildik .Bu methodlar kendi içerisinde updatead methodlarını çağırarak kendi içinde yazan metinleri de güncelledi. Böylece arayüzde dinamik bir yapı oluşturduk.

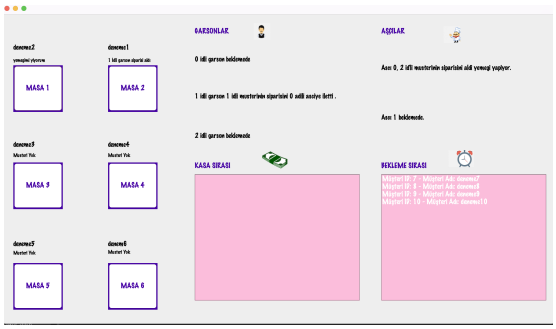


Fig. 6. Projenin Ana Arayüzü

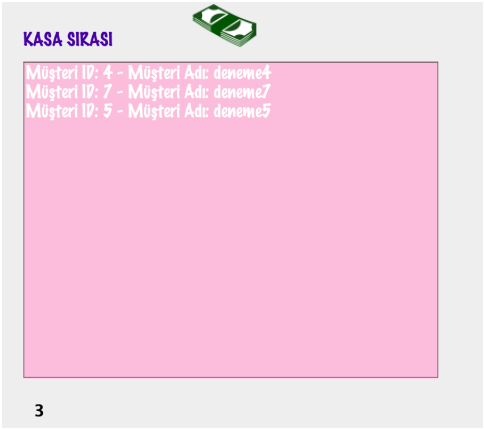


Fig. 7. Kasa Sırası ve Ödeme Yapan Müşterilerin Sayacı

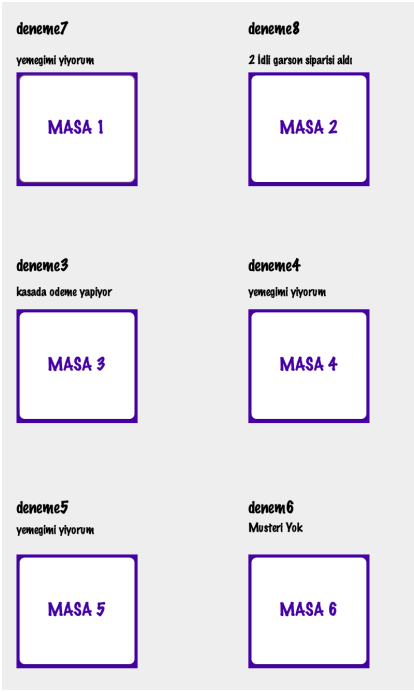


Fig. 8. Masaların ve Masalardaki Müşterilerin Durumları

TXT YAZMA

Oluşturduğumuz dosyayazici classında bir public method tanımladık. Bu method adlığı public ifadeyi belirttiğimiz dosyaya satır satır ekleme işlemi yapıyot. İstediğimiz threadten bu methodu çağırarak belirlediğimiz dosyaya yazma işlemini gerçekleştirdik. Fakat senkronizasyon sıkıntısı yaşamamak için aynı anda sadece bir threadin erişmesi için bu classa locklama işlemi uyguladık.

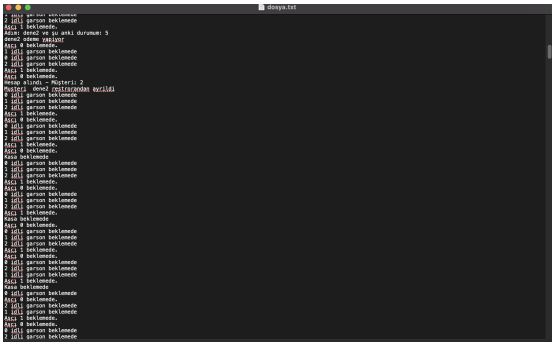


Fig. 9. Txtye Yazdırmamız

!!! ascı,garson,kasa ve musteri sınıfları Thread sınıfından kalıtılmıştır. Masa ve dosyayazıcı sınıfında her hangi bir kalıtım işlemi yapılmamıştır. Arayüz sınıflarımız da JFrameden kaltılmıştır.

PROBLEM 2

Problem 2 için oluşturduğumuz algoritma arayüzde belirtilmiştir.

Fig. 10. Problem 2 nin Arayüze Yansıtılması

III. KATKILAR

Grup olarak projenin her yerinde iletişim halindeydik , birbirimize neler yaptığımızı anlattık.Fakat birimiz daha çok front-end birimizse daha çok back-end taraflarıyla ilgilendik.

IV. KAYNAKÇA

https://stackoverflow.com/questions/299495/how-to-add-an-image-to-a-jpanel

https://www.youtube.com/watch?v=r$_M bozD32eot = 3s$

https://medium.com/sıfırdan-ileri-düzeye-java-eğitim-serisi/multithreaded-programlama-1-kısım-40904a219a18

https://medium.com/@MrAndroid/java-multithreading-part-4-synchronization-11957ccb7b48

V. AKIŞ ŞEMASI SONRAKI SAYFALARDADIR