BLG 503 PARALEL PROGRAMLAMA

PROJE RAPORU

BAHAR-2018

NILAY TUFEK 504151519

Contents

1. Giriş

Bu çalışma BLG 503 Paralel Programlama dersinin 2018 bahar dönemi projesi için yapılmıştır. Ödevde amaç bir banka numara verme uygulaması yapmaktır. Bunun için 4 farklı parallel çalışan bankacı ve bir müşteri sistemi parallel işlemleri oluşturulmuş ve ödevde belirlenen user story'lere göre tasarım ve algoritma gerçeklenmiştir. Dahası "unit test" ler ile sistemin testi de yapılabilmektedir.

Ödev C# dilinde .net framework yardımı ile yazılmış olup, istenilen şekilde programlanmıştır. Paralelleşme için C# thread yapısı kullanılmıştır.

Bankacı sayısı n=4, oturak sayısı m=10 alınmıştır.

Kodu çlaıştırmak için solution'ı vs2017'de açıp run etmek yeterlidir. Test caseler için ise yine VS2017'de Test -> Run All Tests demek yeterlidir.

2. Fonksiyonel ve Non-Fonksiyonel İsterler

2.1 Fonksiyonel İsterler

Ödevde isterler şu şekilde belirlenmiştir.

- Bankacılar paralel çalışan program birimleri olmalıdır
- Müsteri ayrı bir paralel program olmalıdır
- Müşteriler ayrıcalıklı ve normal müşteri olarak ikiye ayrılmakta olup, şu kısıtlara göre bankacılar tarafından çağırılmalıdır:
  1. Hiç ayrıcalıklı müşteri yok ise, sıradaki müşteriyi seç.
  2. Bir ayrıcalıklı müşteriye, kendinden once gelen 2 normal müşteriye göre öncelik tanı.
  3. Daha önce gelmiş ve beklemekte olan normal müşteri var ise, ard arda ikiden fazla ayrıcalıklı müşteriye hizmet verme.
- Gelen müşterilerin sıraya girmesi dilin el verdiği bir "lock" mekanizması ile olmalıdır.
- Bankacıların müşteriyi çağırma ve control etme mekanizmaları ile yine karşılıklı dışlama ile belirlenen kısıtlar çerçevesinde kuyruktan alınmalı ve kuyruktan müşterinin eksilmesi yine "lock" mekaniması ile olmalıdır.
- Ölümcül kilitlenme olmamalıdır
- Sonlu bir program olmalıdır

Eklenecek bir "Unit Test" projesi ile mevcut proje test edilmiştir. Test Driven Development (TDD) yaklaşımının da bir gereği olan, önce unit testlerin yazılımı ve sonrasında kodun çalışması prensibine de bağlı kalınmıştır.

2.2 Non-Fonksiyonel İsterler

Visual Studio 17

.NET 4 veya üzeri gerekmektedir.

3. Tasarım

Tasarımı anlatmak için, sözde kod, akış diyagramı ve sequence diagramlar aşağıdaki gibi hazırlanmıştır:

3.1 Sözde Kod

MüşteriBeklemeListesi'nden okuma ve yazma için iki farklı dışlama objesi gerekmektedir. Bunun nedeni bankacıların kendi aralarında okuma dışlamaları yapmaları ve yazmak için ise ayrı bir dışlamanın kullanılmasıdır. Bankacı okurken müşteri yazmamalıdır. Bu durumda iki farklı mutex objesi kullanılır: mutex_oku ve mutex yaz.

```
Ana_Program()
{
    BankaciArray[4] = Yarat_4_adet_bankaci_paralel_programı(BankciDavranisi)
    Musteri = Yarat_1_adet_musteri_paralel_programı(MusteriDavranisi)

    Bekle_Tum_Paralel_İslemler_Bitene_Kadar();
}
```

```
BankciDavranisi()
{
        while(true)
        {
                if (MusteriGelmis)
                {
                        Lock(mutex_oku)
                        m=SiradakiMusteriyiAl()
                        alinanMusteriListesi(m.id)
                        bekleyenMusteriListesi.cikar(m)
                        UnLock(mutex_oku)
                }
        }
}
```

```
MüsteriDavranisi()
{
        while(true)
        {
                if (yeniMusterGelmis)
                {
                        Lock(mutex_yaz)
                        BekleyenMusteriListesineEkle(m)
                        UnLock(mutex_yax)
                }
        }
}
```

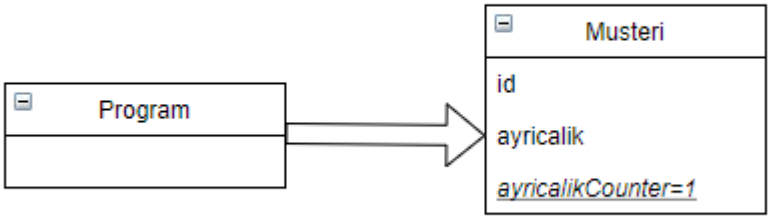
```
BekleyenMusteriListesineEkle(musteri m)
{
        if (bekleyenMusteriListesi.Count == beklemeKoltugu + 5)
        {
                Print("Şu an doluyuz, sonra geliniz");
```

```
                return;
        }
        Print("Added: " + musteri.id.ToString());
        bekleyenMusteriListesi.AddLast(musteri);
}
```

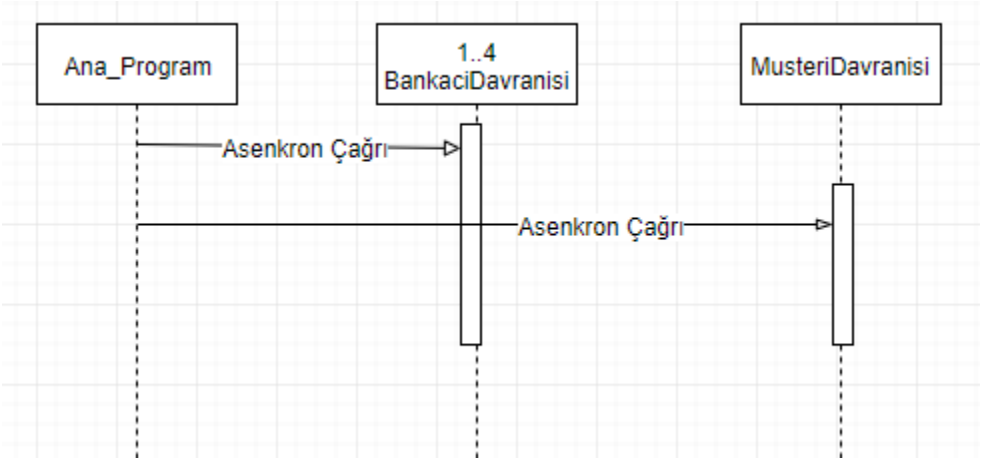
```
SiradakiMusteriyiAl(): Musteri
{
        var node = bekleyenMusteriListesi.First;
        if (node == null)
        {
                return null;
        }
        var m1 = bekleyenMusteriListesi.ElementAt(0);
        //birinci sıradaki müsteri ayrıcalıklı ise onu al, sıradan çıkar
        if (m1.ayricalikli)
        {
                bekleyenMusteriListesi.Remove(node);
                return m1;
        }
        //bir kişi bekliyorsa birinci kişiyi al
        if (bekleyenMusteriListesi.Count < 2)
        {
                bekleyenMusteriListesi.Remove(m1);
                ayricalikCounter = 0;
                return m1;
        }

        var m2 = bekleyenMusteriListesi.ElementAt(1);
        node = node.Next;
        // bekleyen 2. kişi ayrıcalıklı ve
        // kendinden önce 2 ayrıcalıklı bu hakkı kullanmamışsa
        // 2.yi al
        if (m2.ayricalikli && ayricalikCounter < 2)
        {
                bekleyenMusteriListesi.Remove(node);
                ayricalikCounter++;
                return m2;
        }
        if (bekleyenMusteriListesi.Count < 3)
        {
                bekleyenMusteriListesi.Remove(m1);
                ayricalikCounter = 0;
                return m1;
        }

        node = node.Next;
```

```
            var m3 = bekleyenMusteriListesi.ElementAt(2);
            // bekleyen 3. kişi ayrıcalıklı ve
            // kendinden önce 2 ayrıcalıklı bu hakkı kullanmamışsa
            // 3.yü al
            if (m3.ayricalikli && ayricalikCounter < 2)
            {
                  bekleyenMusteriListesi.Remove(node);
                  ayricalikCounter++;
                  return m3;
            }
            //listede bekleyen ilk 3 kişiden kimse ayrıcalıklı değilse
            //ya da ard arda iki ayrıcalık hakkını kullanmışsa
            //ilkini al
            bekleyenMusteriListesi.Remove(m1);
            ayricalikCounter = 0;
            return m1;
}
```

3.2 Sınıf Diagramı



3.3 Sequence Diagram



4. Algoritma, Veri Tipleri ve Karşılaşılan Zorluklar

Algorithma genel olarak 2 farklı parallel işlem yapısı üzerine oturmaktadır. Bir tip bankacılık işlemleri (ki bunalrdan 4 tane var) ikincisi ise müşteri bekleme sırasına grime işlemleri için. Müşteriler geliş sıralarına göre BeklemeListesine girerler, Bankacılar ise bu listeden ayrıcalık ve kurallara göre kişileri seçerler. 15' kişiden fazla bekleyeyen olamayaağı için onlar daha sonar gelmek üzere sıra numarası alamadan ayrılırlar. (ID atanmaz)

Tüm müşteriler "Musteri" Sınıfının nesneleridirler. Her birinin "ID" ve "ayrıcalık" özellikleri mevcuttur. ID numarası geliş sırasını belirtir. Ayrıcalıklı olanlar için ise ayrıcalık bilgisi "true" diğerleri için "false"'tur.

Listeler bağlı-liste tipinde tutulmuş olup, ekleme çıkarma işlemlerinde kolaylık sağlamıştır.

BeklemeListesi'nden kurallara göre alınanlar "Alınmışlar" listesine eklenirler ve böylece testler control edilebilir.

Burada parallel işlemlerin yapımında zorluklar dil kısıtları ve işlemci hızından bağımsız olarak "yarışma durumu" için vardır. Bunun için yazarken ve okurken (Okuduktan sonar silinme durumundan) dolayı karşılıklı dışlama gerekmiştir. Böylece C# dilinde "Lock" mekanizması ile bu sağlamıştır.

5. Test Senaryoları ve Sonuçlar

5.1 Dort Adet Paralel Bankacılık İşlemi:

```
Bankaci PID: 3 waiting
Bankaci PID: 4 waiting
Bankaci PID: 5 waiting
Bankaci PID: 6 waiting
```

4 adet banacının PID'leri 3,4,5 ve 6 imiş.

5.2 Paralel Musteri İşlemi

```
Bankaci PID: 3 waiting
Bankaci PID: 4 waiting
Bankaci PID: 5 waiting
Bankaci PID: 6 waiting
Bankaci PID: 3 waiting
Bankaci PID: 6 waiting
Bankaci PID: 4 waiting
Bankaci PID: 5 waiting
Bankaci PID: 4 waiting
Bankaci PID: 6 waiting
Bankaci PID: 5 waiting
Bankaci PID: 3 waiting
Added: 1
```

Bankacilar meşgul beklemede iken Musteri process'I 1 no'lu ID li müşteryi eklemiştir. Toplam 5 adet parallel işlem vardır. Nir de ana thread olduğu için hepsi beraber 6 paralel işlem vardır.

(1 ana thread, 4 paralel bankacı threadi bir de müsterileri alan thread)

5.3 On Beş Kişiden Fazla Musteri Gelme Durumu

```
Bankaci PID: 4 waiting
Bankaci PID: 3 waiting
Bankaci PID: 5 waiting
Bankaci PID: 6 waiting
Bankaci PID: 6 waiting
Bankaci PID: 3 waiting
Bankaci PID: 5 waiting
Bankaci PID: 4 waiting
Added: 1
Added: 2
Added: 3
Added: 4
Added: 5
Added: 6
Added: 7
Added: 8
Added: 9
Added: 10
Bankaci PID: 4 isleme aldigi musteri ID: 2
Added: 11
Added: 12
Added: 13
Added: 14
Added: 15
Added: 16
Su an doluyuz, sonra geliniz
```

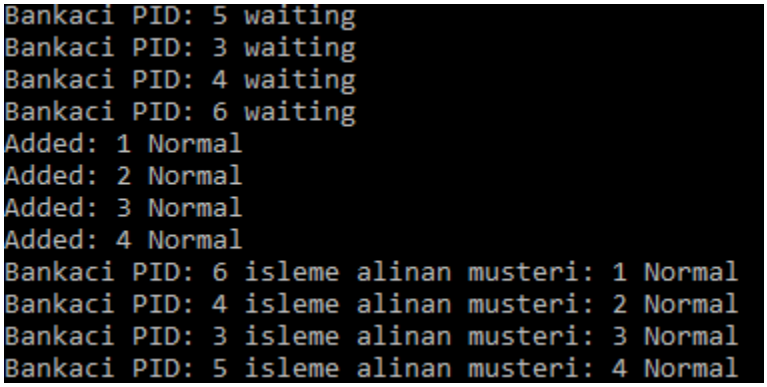
Bu örnekte 1 numaralı müşteri “normal”, 2 numaralı müşteri “ayrıcalıklı” dir. Öncelikle bankacılık işlemleri parallel çalışıp meşgul beklemede iken müşteriler gelmeye başlamış ve 4 numraları PID’ye sahip olan bankacı bankada 10 tane müşteri beklerken, “ayrıcalıklı olan” 2 numaralı müşteriyi işleme almış ve bekleyen sayısı 9 a düşmüşken 6 tane daha beklemeye kişi alınmıştır. Toplam bekleyen sayısı 15 iken 1 müşteri daha gelmeye çalışmış fakat “Şu an doluyuz, sonar geliniz” mesajı ile karşılaşılmıştır.

5.4 Ayrıcalık Durumuna Göre Senaryolar

5.4.1 Normal-Normal-Normal-Normal Durumu:

BeklemeSırası 1-2-3-4 iken

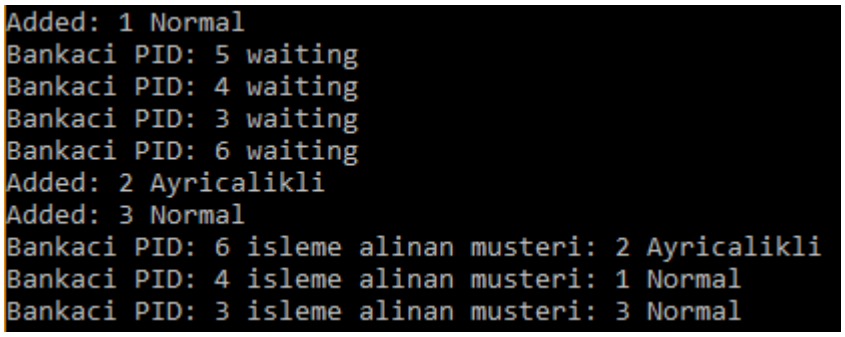
İşlemeAlınanSıra: 1-2-3-4 olmalıdır



5.4.2 Normal-Ayrıcalıklı-Normal Durumu:

BeklemeSırası 1-2-3 iken

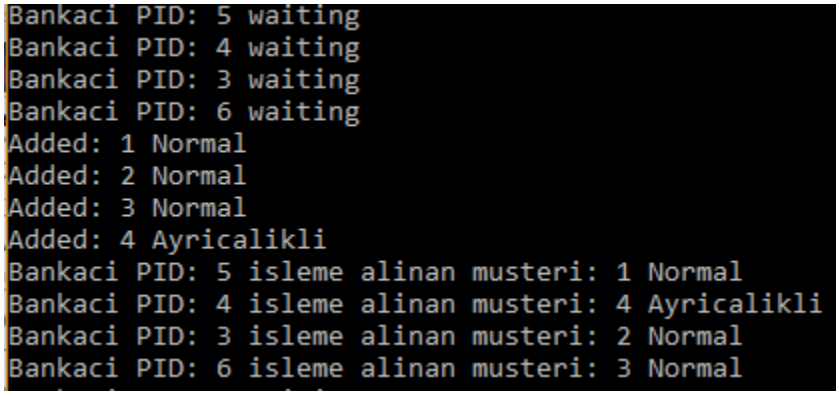
İşlemeAlınanSıra: 2-1-3 olmalıdır



5.4.3 Normal-Normal-Normal-Ayrıcalıklı Durumu:

BeklemeSırası 1-2-3-4 iken

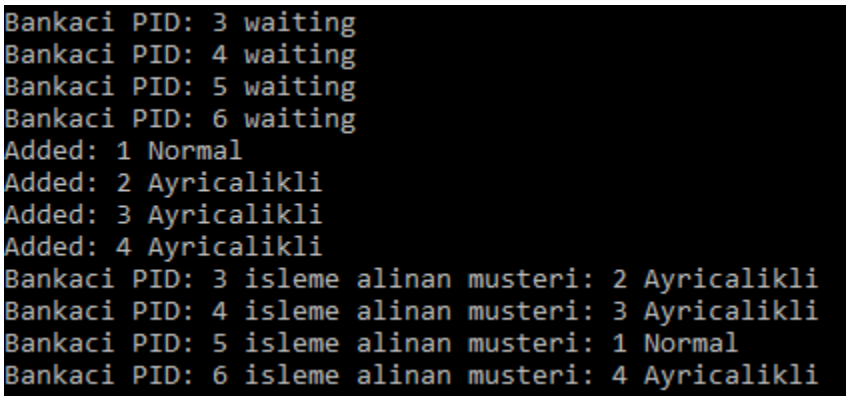
İşlemeAlınanSıra: 1-4-2-3 olmalıdır



5.4.4 Normal-Ayrıcalılı-Ayrıcalıklı-Ayrıalıklı Durumu:

BeklemeSırası 1-2-3-4 iken

İşlemeAlınanSıra: 2-3-1-4 olmalıdır



Görüldüğü üzere tüm kritik testleri geçmiştir.

Sonuç

Ödev başarı ile gerçeklenmiş ve user storylere göre oluşturulan testleri geçmiştir.