ASANSÖRLERDEKİ TALEP YOĞUNLUĞUNUN MULTITHREAD İLE KONTROLÜ

Sefa Mert GÜNGÖR
Bilgisayar Mühendisliği Bölümü
Kocaeli Üniversitesi
180201086

Miraç Onur SAKA
Bilgisayar Mühendisliği Bölümü
Kocaeli Üniversitesi
180201081

ÖZET- Bu projede temel amaç threadler kullanılarak multithread kavramını asansörlerin çalışma mantığıyla harmanlayarak projeyi gerçekleştiren kişiler tarafından bu kavramı tamamiyle öğrenmektir. Buna bağlı olarak bizden bir alışveriş merkezindeki asansörlerde talep yoğunluğuna bağlı olarak bir uygulama geliştirmemiz istenmektedir. Bunlara bağlı olarak multithread yapısı kullanılarak bir asansör projesi gerçekleştirilmiştir.

Anahtar kelimeler-

Asansör, thread, multithread, eş zamanlılık, talep yoğunluğu

I.GİRİŞ

Projemizde birden çok thread kullanılarak multithread yapısına sahip bir alışveriş merkezindeki asansörleri programlamış olduk. Öncelikle 5 asansör threadi, giriş threadi, çıkış threadi, kontrol threadi olmak üzere toplamda 8 ayrı runnable class oluşturduk. 5 adet asansör threadinin kuyruk bilgilerini taşıyacak olan 2 boyutlu bir arraylist tanımladık. Daha sonra içini dolduruğumuz 2 boyutlu bu arraylist'deki bilgileri kullanarak multi thread projemizi gerçekleştirmiş olduk. Bu projede thread, multithread,arraylist yapılarının ve random sınıfının bir arada kullanımına yönelik bir çalışma gerçekleştirilmiştir.

II.HAZIRLIKLAR VE BİLGİLER

Bu proje Java programlama dilinde geliştirilmiş olup, geliştirme ortamı olarak "Apache Netbeans 11.1" kullanılmıştır. İlk önce bizden istenen durumları değerlendirip tartışarak ne yapmamız gerektiği hakkında karara varıp projeyi hangi adımlara göre takip edeceğimize karar verdik. Daha sonrasında projeye başladık.

III.YÖNTEM

Projemizde ilk önce multithread kavramını içinde kullanılan fonksiyonları, senkronizasyon durumlarını, random sınıfının nasıl kullanıldığı hakkında araştırmalar yaparak bilgi edinmiş olduk. Daha sonra ilk olarak giriş threadini oluşturup tamamladık. Giriş threadinde her

500 ms zaman aralıklarıyla 1-10 arasında rastgele sayıda müşterinin avmye giriş yapmasını sağlayan algoritmayı yazdık. Böylelikle giriş threadini tamamladık. Daha sonra bizden istenildiği gibi 1000 ms zaman aralıklarıyla 1-5 arasında rastgele sayıda müşterinin avmden çıkış yapmasını sağlayan çıkış threadi algortimamızı yazmış olduk. Bu thread çıkmak isteyen müşterileri rastgele bir kattan (1-4) zemin kata gitmek için asansör kuyruğuna alır. Daha sonra işlevleri aynı olan, katlardaki kuyrukları kontrol eden, maksimum kapasiteyi aşmayacak şekilde (10) kuyruktaki müşterilerin talep ettikleri katlara taşınabilmesini sağlayan 5 adet asansör threadi algoritmasını yazmış olduk. Daha sonra katlardaki kuyrukları kontrol eden kuyrukta bekleyen kişilerin toplam sayısı asansörün kapasitesinin 2 katını (20) aştığı durumda yeni asansörü aktif hale getiren, kuyrukta bekleyen kişilerin toplam sayısı asansör kapasitesinin altına indiğinde ise asansörlerden birini pasif hale getiren kontrol threadi algoritmasını yazmış olduk. Son olarak bu bilgileri terminale 200 ms aralıklarıyla yazdırdık. Bu şekilde projemizi tamamladık.

Bu projede kullanılan sınıflar:

1)Asansör Sınıfları

- **MainAsansor**
- **Asansor2**
- **Asansor3**
- **Asansor4**
- **Asansor5**

2)Kontrol

3)Giris

4)Cikis

5)Bilgi

6)Ana

Bu başlık altında sınıflar ayrı ayrı ele alınacak ve detayları anlatılacaktır.

1)Asansor Sınıfları

Katlardaki kuyrukları kontrol eder. Maksimum kapasiteyi aşmayacak şekilde kuyruktaki müşterilerin talep ettikleri katlara taşınabilmesini sağlar. Zemin kattan diğer katlara gitmek isteyen müşterileri ve diğer katlardan müşterileri kuyruktan alır.

2) Giris

Bu sınıf 500 ms zaman aralıklarıyla (1-10) arasında rastgele sayıda müşterinin AVM'ye giriş yapmasını sağlamaktadır (Zemin Kat). Giren müşterileri rastgele bir kata (1-4) gitmek için asansör kuyruğuna alır.

3)Cikis

1000 ms zaman aralıklarıyla (1-5) arasında rastgele sayıda müşterinin AVM'den çıkış yapmasını sağlamaktadır (Zemin Kat). Çıkmak isteyen müşterileri rastgele bir kattan (1-4), zemin kata gitmek için her kat için ayrı oluşturulan asansör kuyruğuna alır.

4)KontrolThread

Katlardaki kuyrukları kontrol eder. Kuyrukta bekleyen kişilerin toplam sayısı asansörün kapasitesinin 2 katını aştığı durumda (20) yeni asansörü aktif hale getirir. Kuyrukta bekleyen kişilerin toplam sayısı asansör kapasitenin altına indiğinde asansörlerden biri pasif hale gelir. Fakat

ana asansör her zaman çalışmaya devam eder.

5)Bilgi

Bütün threadlerin kullanığı ortak bilgileri içinde bulunduran sınıftır. Katlardaki kuyrukların ve kişi sayısının bilgisini tutar. Bu sayede threadlerin iletişim kurmasını kolaylaştırır.

6)Ana

Tüm sınıfların çağrıldığı ve thread'lerinin çalıştırıldığı sınıftır. Bütün threadlerin işlemleri 200 ms de bir "While" döngüsü ile konsola bu sınıfta yazdırılır. Proje bu sınıf üzerinden çalıştırılır.

5)Cikis ile alışveriş merkezinden çıkış yapmak isteyen müşteriler bulundukları katların asansör kuyruklarına ekledi.

6)Kontrol kuyruklardaki yoğunluğu kontrol etti ve kuyruk sayısı asansör kapasitesinin 2 katına çıktığında yeni bir asansörü aktif etti. Azaldığında ise pasif hale getirdi.

7)200 ms'de bir asansörlere, katlara ve müşteri durumlarına ait bilgiler konsol ekranına yazdırıldı. **ek1 *ek2 *ek3*

****Ana isimli sınıfta 200ms'de bir konsola bütün bilgilerin yazdırılmasını sağladık. Bunun sebebi asansörler 200 ms'de bir hareket ettikleri için asansörlerin hareketlerini doğru bir şekilde gözlemleyebilmeyi sağlamaktır. Fakat bu durum diğer thread çalışmaları değişmeden de tekrar tekrar aynı bilgilerin yazılmasına neden oldu.**

IV.YALANCI KOD

1)Program çalıştı.

2)Tüm thread'ler eş zamanlı olarak çalış-maya başladı.

3)Giris çalışarak giriş katın asansör kuyruğuna müşterileri ekledi.

4)Asansörler müşterileri gitmek istedikleri katlara çıkardı ve çıkış yapmak isteyen müşterileri zemin kata indirdi.

V.EKRAN GÖRÜNTÜLERİ

*ek1

```
KATLARDAKI KISI SAYISI
0. Kat Kisi Sayisi : 14 Kuyruk Saysisi : 14
1. Kat Kisi Sayisi : 24 Kuyruk Saysisi : 0
2. Kat Kisi Sayisi : 0 Kuyruk Saysisi : 0
3. Kat Kisi Sayisi : 0 Kuyruk Saysisi : 0
4. Kat Kisi Sayisi : 15 Kuyruk Saysisi : 4
Çıkış sayacı : 4
```

*ek2

```
ANA ASANSOR
Active : true
                mode : working
                Insan Sayisi : 5
                Bulundugu Kat : 1
                Icindekiler : [4, 4, 4, 4, 4]
                Hedef : 4
                Direction : Up
2.ASANSOR
Active : false
                mode : idle
                Insan Sayisi : 0
                Bulundugu Kat : 3
                Icindekiler : []
                Hedef : 0
                Direction : Down
3.ASANSOR
Active : false
                mode : idle
                Insan Sayisi : 0
                Bulundugu Kat : 0
                Icindekiler : []
                Hedef : 0
4.ASANSOR
Active : false
                mode : idle
                Insan Sayisi : 0
                Bulundugu Kat : 0
                Icindekiler : []
                Hedef : 0
5.ASANSOR
Active : false
                mode : idle
                Insan Sayisi : 0
                Bulundugu Kat : 0
                Icindekiler : []
                Hedef : 0
```

*ek3

```
0. Kat : [[1,1][4,2][9,4]]
1. Kat : []
2. Kat : []
3. Kat : []
4. Kat : [[4,0]]
```

VI.SONUÇ

Oluşturulan algoritmaların eş zamanlı bir şekilde çalışması için bunları birleştirerek bir program geliştirilmiştir. İşletim Sistemleri dersinde öğrenilen Thread yapısı bu proje sayesinde pekiştirilmiştir.

VII.REFERANSLAR

1- https://stackoverflow.com/

2- https://gelecegiyazanlar.turkcell.com.tr/

3- https://www.udemy.com/

4- https://www.geeksforgeeks.org/

5- https://medium.com/