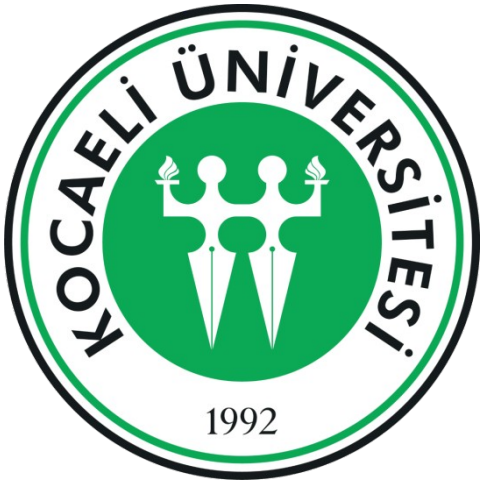


KOCAELİ ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ

YAZILIM LABORATUVARI-1 PROJE -2

ASANSÖRLERDEKİ TALEP YOĞUNLUĞUNUN MULTITHREAD İLE KONTROLÜ

ENGİN YENİCE
190201133

[RAPOR]

KOCAELİ 2020

ASANSÖRLERDEKİ TALEP YOĞUNLUĞUNUN MULTITHREAD İLE KONTROLÜ

Engin Yenice
Bilgisayar Mühendisliği
Kocaeli Üniversitesi
190201133
enginyenice2626@gmail.com

***Özet*— AVM'deki asansörlere gelen isteklerdeki yoğunluğu, multithread kullanarak diğer asansörlerle birlikte azaltmaktır.**

Keywords—Asansör,Thread, Kat, Avm, Kuyruk, Multithread

I. GİRİS

Proje C# Programlama dili kullanılarak nesne tabanlı programlama mantığı ile geliştirildi. Projenin amacı bir AVM'deki asansörlere gelen isteklerde ki yoğunluğu, multithread kullanarak diğer asansörlerle birlikte azaltmaktır.

AVM'nin kendisine ait belirli özellikleri vardır.

- AVM'deki kat sayısı 5'tir. (Zemin kat, birinci kat, ikinci kat, üçüncü kat ve dördüncü kat)
- Toplamda 5 adet asansör bulunmaktadır.
- Asansörlerin biri sürekli çalışmaktadır. Geriye kalanlar, yoğunluk durumuna göre aktif veya pasif durumdadır.
- Asansörlerin maksimum kapasitesi 10'dur.
- Asansörlerdeki kat arası geçiş 200 ms'dir.

Ayrıca AVM giriş, çıkış, asansör ve kontrol işlemlerin belirli şartlara göre yapılmaktadır. Bu şartlar:

AVM Giriş (Login) Thread: 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.

AVM Çıkış (Exit) Thread: 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 asansör kuyruğuna alır.

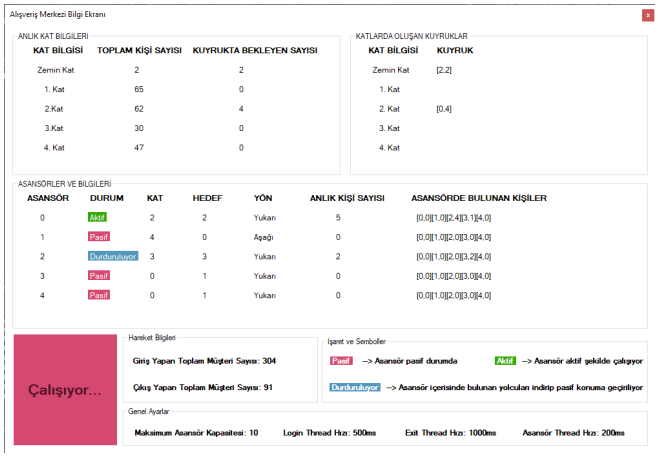
Asansör Thread: Katlardaki kuyrukları kontrol eder. Maksimum kapasiteyi aşmayacak şekilde kuyruktaki müşterilerin talep ettikleri katlarda taşınabilmesini sağlar. Bu thread asansör sayısı kadar (5 adet) olmalıdır.

NOT: Zemin kattan diğer katlara (AVM'ye) giriş yapmak isteyenler, ya da diğer katlardan (AVM'den) çıkış yapmak isteyenler kuyruk oluştururlar.

Kontrol Thread: 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. Bu işlem tek asansörün çalıştığı durumda geçerli değildir.

II. YÖNTEM

A. Form Ekranı



Şekil 1 : Form Ekranı

Proje tek bir ekran üzerinden yönetilmektedir. Form ekranının görüntüsü **Şekil 2** de belirtilmiştir. Form ekranı toplamda 7 parçaya bölünerek bilgileri göstermektedir. Yukarıdan aşşağıya doğru bu parçaları inceleyecek olursak.

B. ANLIK KAT BİLGİLERİ

ANLIK KAT BİLGİLERİ

KAT BİLGİSİ	TOPLAM KİŞİ SAYISI	KUYRUKTA BEKLEYEN SAYISI
Zemin Kat	2	2
1. Kat	65	0
2.Kat	62	4
3.Kat	30	0
4. Kat	47	0

Şekil 2

Bu bölümde katlarda bulunan toplam kişi sayısını ve kuyrukta bekleyen kişi sayısını göstermektedir.

C. KATLARDA OLUŞAN KUYRUKLAR

Şekil 3

Bu bölümde o kattan başka bir kata geçmek isteyen müşterilerin kuyruk sıralamaları gösterilmektedir. Kuyruk gösteri [**hedef kat**, **gidecek müşteri sayısı**]

Örnek olarak Zemin katın kuyruğunu incelersek:

[2,2] => 2. Kata 2 müşteri gitmek için kuyruğa girmiş.

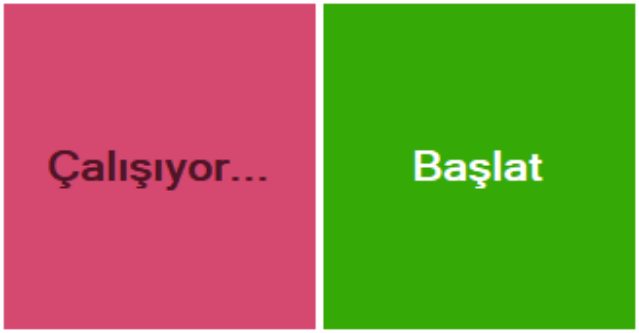
toplamda 2 müşteri kuyrukta sıra beklemektedir. Hesaplamanın doğruluğunu **Katlarda oluşan kuyruklar** başlığı altında bulunan **Şekil 2** üzerinden kontrol edebilirsiniz.

D. ASANSÖRLER VE BİLGİLERİ

Şekil 4

Bu bölüm üzerinden her bir asansörün ismini, modunu, bulunduğu katı, hedeflediği katı, hareket yönünü, anlık olarak içerisinde bulunan kişi sayısını ve katlara gitmek isteyen müşterilerin detaylı bilgilerini görüntüleyebilirsiniz. Asansörlede bulunan kişilerin gösterim [**hedef kat**, **gidecek müşteri sayısı**] şeklinde gösterilmektedir. Durum özellikleri hakkında detaylı bilgi **İŞARET VE SEMBOLLER** başlığı altında anlatılmaktadır.

E. SİSTEM BUTONU



Şekil 5

Form içerisinde tek bir buton bulunmaktadır. Bu buton sistemi başlatmaktadır. Sistem başlatılması durumunda kapatılanadek durmamaktadır.

Eğer sistem başlatılmamış ise **yeşil renkte Başlat** butonu gösterilmektedir.

Eğer sistem başlatılmış ise **kırmızı renkte Çalışıyor** butonu gösterilmektedir.

F. HAREKET BİLGİLERİ

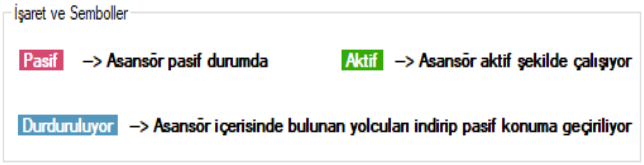
Şekil 6

Sistem üzerinden anlık olarak hareket bilgilerini göstermektedir.

Giriş Yapan Toplam Müşteri Sayısı: Sistemin çalıştığı süre boyunca AVM giriş yapan toplam müşteri sayısını gösterir.

Çıkış Yapan Toplam Müşteri Sayısı: Sistemin çalıştığı süre boyunca AVM den çıkış yapan toplam müşteri sayısını gösterir.

G. İŞARET VE SEMBOLLER



Şekil 7

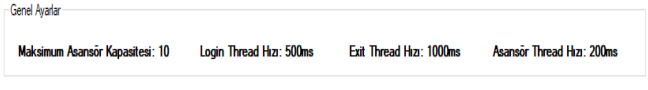
Asansörlerin durumunu göstermektedir. Sistem içerisinde temel olarak 2 durum vardır. Fakat asansörün yolcu taşıdığı durumlarda pasif konuma geçmesi gerekiyorsa eğer içerisinde bulunan yolcuları gerekli katlarına bırakıp pasif duruma geçmektedir.

Aktif: Asansör çalışıyor.

Pasif: Asansör çalışmıyor.

Durduruluyor: Asansör içerisinde bulunan yolcuları indirip **Pasif** konuma geçmeye hazırlanıyor. Bu durumda olan bir asansör sadece asansörden müşteri indirme işlemi yapmaktadır. Asansör içerisine yeni müşteri eklemez.

H. GENEL AYARLAR



Şekil 8

AVM nin genel ayarlarının belirtildiği bölümdür. Belirtilen parametreler dinamik olarak yazılmaktadır.

III. KLASÖR YAPISI

Projenin yönetilmesi açısından proje temel olarak **Tasarım** ve **Yönetim** olmak üzere 2 yapıya bölünmüştür.

Projenin tüm yönetimsel işlemleri **ShoppingCenter** klasörü içerisinde ve tasarımsal yapısı **UI** klasörü içerisinde gerçekleşmektedir.

```
Talep_Yogunlugunun_Multithread_Kontrolu
├── UI/
│   └── ShoppingMallInformationDisplay.cs
└── ShoppingCenter/
        ├── Core/
        │       └── Settings.cs
        ├── Elevator/
        │       ├── Abstract/
        │       │       └── IElevator.cs
        │       └── Concrete/
        │               └── Elevator.cs
        ├── Floor/
        │       ├── Abstract/
        │       │       └── IFloor.cs
        │       └── Concrete/
        │               └── Floor.cs
        └── Threads/
                ├── Abstract/
                │       ├── ITControl.cs
                │       ├── ITElevator.cs
                │       ├── ITExit.cs
                │       └── ITLogin.cs
                └── Concrete/
                        ├── TControl.cs
                        ├── TElevator.cs
                        ├── TExit.cs
                        └── TLogin.cs
```

Şekil 9

IV. PROJEDE BULUNAN SINIFLAR VE FONKSİYONLARI

Proje içerisinde class, interface tipinde sınıflar kullanılmıştır.

A. Interface

Nesne yönelimli programlama dillerinde arayüz, değişik sınıflardan nesnelerin kategorize edilmesini sağlayan bir soyut tür çeşididir.

B. Class

Sınıf, nesne yönelimli programlama dillerinde nesnelerin özelliklerini, davranışlarını ve başlangıç durumlarını tanımlamak için kullanılan şablonlara verilen addır. Bir sınıftan türetilmiş bir nesne ise o sınıfın örneği olarak tanımlanır. Sınıflar genelde şahıs, yer ya da bir nesnenin ismini temsil ederler.

C. ShoppingCenter/Core/Settings.cs (Class)

Genel ayarların bulunduğu sınıf.

int Capacity { get; }: Tüm asansörlerin kapasitesini belirtir.

int ElevatorSpeed { get; }: Asansör(Elevator) thread hızını belirler.

LoginSpeed { get; }: Giriş(Login) threadının hızını belirler.

ExitSpeed { get; }: Çıkış(Exit) threadının hızını belirler.

int TotalLoginCount {get; set;}: Alışveriş merkezine giren toplam müşteri sayısını günceller ve döndürür

int TotalExitCount {get; set;}: Alışveriş merkezinden çıkan ve çıkmak için kuyruğa giren toplam müşteri sayısını günceller ve döndürür

int TotalLogoutCount {get; set;}: Alışveriş merkezinden çıkan toplam müşteri sayısını günceller ve döndürür

Settings(): Alışveriş merkezinin genel ayarlarının tutulduğu yapıcı metot.

D. ShoppingCenter/Elevator/Abstract/IElevator.cs (Interface)

Elevator sınıfını kategorize etmek için kullanılan bir sınıftır. Elevator sınıfı içerisinde bulunması gerekli public metotları belirtmektedir.

E. ShoppingCenter/Elevator/Concrete/Elevator.cs (Class)

int Name { get; set; }: Asansörün isminin güncellenmesi ve getirilmesi işlemini yapar.

bool IsActive { get; set; }: Asansörün çalışıp çalışmadığının bilgisinin güncellenmesi ve getirilmesi işlemini yapar.

int Destination { get; set; }: Asansörün gitmek için hedeflediği katın bilgisinin güncellenmesi ve getirilmesi işlemini yapar.

bool Direction { get; set; }: Asansörün gidece yönü belirtir ve güncellenmesi ve getirilmesi işlemini yapar.

int Floor { get; set; }: Asansörün bulunduğu katı belirtir ve güncellenmesi ve getirilmesi işlemini yapar.

void SetFloorCount(int floor, int count): Asansörde bulunan müşteri sayısını günceller.

string FloorCountString(): Asansörde bulunan müşterileri belirli bir formatta string olarak geri döndürür

int GetFloorCount(int floor): Parametre olarak gönderilen katta inecek kişi sayısını geriye döndürür.

void FloorCountClear(): Asansörde bulunan tüm müşterileri temizler.

int GetCount(): Asansör içerisinde kaç adet müşteri olduğunu döndürür.

int GetFirstDestination(): *Bulunduğu kata en yakın hedefi belirler.*

F. ShoppingCenter/Floor/Abstract/IFloor.cs (Interface)

Floor sınıfını kategorize etmek için kullanılan bir sınıftır. *Floor* sınıfı içerisinde bulunması gerekli public metotları belirtmektedir.

G. ShoppingCenter/Floor/Concrete/Floor.cs (Class)

int Name { get; set; }: Alışveriş merkezinde bulunan katın isminin güncellenmesi ve getirilmesi işlemini yapar.

int FloorCount { get; set; }: Katta bulunan müşteri sayısının güncellenmesi ve getirilmesi işlemini yapar.

int QueueCount { get; set; }: Katın kuyruğunda bulunan müşteri sayısının güncellenmesi ve getirilmesi işlemini yapar.

void RetryQueue(int floor, int count): Kuyruğun başındaki değerin güncellenmesi için kullanılan metot.

void SetFloorQueue(int floor, int count): Kat kuyruğuna yeni eleman ekleme metodu.

void CreateFloorQueue(int floor, int count): Kat kuyruğuna yeni eleman ekleme ve kuyruktaki toplam müşteri sayısını arttırma metodu.

string FloorQueueString(): Katta bulunan müşterileri belirli bir formatta string olarak geri döndürür

Queue<string> GetFloorQueue(): Kuyruğu geriye döndürür.

H. ShoppingCenter/Threads/Abstract/ITControl.cs (Interface)

TControl sınıfını kategorize etmek için kullanılan bir sınıftır. TControl sınıfı içerisinde bulunması gerekli public metotları belirtmektedir.

I. ShoppingCenter/Threads/Abstract/ITElevator.cs (Interface)

TElevator sınıfını kategorize etmek için kullanılan bir sınıftır. TElevator sınıfı içerisinde bulunması gerekli public metotları belirtmektedir.

J. ShoppingCenter/Threads/Abstract/ITExit.cs (Interface)

TExit sınıfını kategorize etmek için kullanılan bir sınıftır. TExit sınıfı içerisinde bulunması gerekli public metotları belirtmektedir.

K. ShoppingCenter/Threads/Abstract/ITLogin.cs (Interface)

TLogin sınıfını kategorize etmek için kullanılan bir sınıftır. TLogin sınıfı içerisinde bulunması gerekli public metotları belirtmektedir.

L. ShoppingCenter/Threads/Concrete/TControl.cs (Class)

bool ControlThread(Floor.Concrete.Floor[] floors, Elevator.Concrete.Elevator[] elevators, int capacity): 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. Bu işlem tek asansörün çalıştığı durumda geçerli değildir.

M. ShoppingCenter/Threads/Concrete/TElevator .cs (Class)

void ElevatorThread(Elevator.Concrete.Elevator elevator, Floor.Concrete.Floor[] floors, int capacity, Settings settings): Katlardaki kuyrukları kontrol eder. Maksimum kapasiteyi aşmayacak şekilde kuyruktaki müşterilerin talep ettikleri katlarda taşınabilmesini sağlar. Bu thread asansör sayısı kadar (5 adet) olmalıdır. NOT: Zemin kattan diğer katlara (AVM'ye) giriş yapmak isteyenler, ya da diğer katlardan (AVM'den) çıkış yapmak isteyenler kuyruk oluştururlar.

void FloorChange(Elevator.Concrete.Elevator elevator): Asansör kat arttırma ve azaltma işlemi

int CheckTopFloor(Floor.Concrete.Floor[] floors, int maxDestinationalFloor): Bulunduğu katın üstündeki katları kontrol eder. Zemin kata inecek müşteri varsa hedef olarak onu belirler

int CheckButtomFloor(Floor.Concrete.Floor[] floors, int elevatorFloor): Bulunduğu katın altındaki katları kontrol eder. Zemin kata inecek Müşteri varsa hedef olarak onu belirler

void PassengerLowering(Elevator.Concrete.Elevator elevator, Floor.Concrete.Floor[] floors, Settings settings): Asansör içerisindeki müşterileri bulundukları katlara geldiğinde indirme işlemini yapar

void PassengerBoarding(Elevator.Concrete.Elevator elevator,Floor.Concrete.Floor[] floors, int capacity): Bulunduğu katın kuyruğunda müşteri varsa müşteriyi asansörün kapasitesine uygun olacak şekilde asansöre alır.

N. ShoppingCenter/Threads/Concrete/TExit .cs (Class)

void ExitThread(Floor.Concrete.Floor[] floors, Settings settings): [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 asansör kuyruğuna alır.

O. ShoppingCenter/Threads/Concrete/TLogin .cs (Class)

void LoginThread(Floor.Concrete.Floor[] floors, Settings settings): [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.

V. KABA KOD

190201133-Kaba-Kod.pdf olarak teslim dosyaları içerisine eklenmiştir.

VI. KAYNAKÇA

- https://enginyenice.com/
- https://stackoverflow.com/
- http://tutorialspoint.com/
- https://medium.com/
- https://social.msdn.microsoft.com/