**T.C.**
**SAKARYA ÜNİVERSİTESİ**
**BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ**
**BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**

**Ders** : **İşletim Sistemleri**

**Şube** : **İkinci Öğretim – B Grubu**

**Dersi Veren** : **Ünal Çavuşoğlu**

**Ödev Konusu** : **Proses Simülasyonu**

**Dönem** : **2022-2023 Güz Dönemi**

**Hazırlayanlar** : **Ahmed Asım Avutmuşlu**
**G211210373**
**Muhsin Karataş**
**B221210502**

**Github Linki** :
**https://github.com/ahmedasimavutmuslu/IsletimSistemleriProje**

## Ödevin Konusu ve Amacı

Yapılan ödevde bizden istenilen derste öğrenilen bilgiler ışığında proseslerin simülasyonunu sağlamaktır. Bu ödev sayesinde öğrendiğimiz şeyleri pekiştirecek ve eksiklerimizi tamamlayacağız.

.

## 1- Ödevde Zorlandığımız Kısımlar

Genel olarak Dispatcher ve Scheduler arasındaki ilişkiyi çözmemiz biraz zaman aldı. Hatta yazdığımız proje de köklü değişikliklere gidilmesine neden oldu. Bunun dışında her prosese renk atanması görevini gerçekleyebilmek için daha önce kullanmadığımız bazı yapıları kullanmak zorunda kaldık. Bunun iyi yanı bizi araştırmaya itmiş olmasıydı.

## 2- Kullandığım Kaynaklar ve Öğrendiklerimiz

Öncelikle çoğunlukla zorlandığımız kısımları Youtube üzerindeki bir çok kanaldan farklı örnekler üzerinden izleyerek öğrenmeye çalıştık. Bunun dışında w3schools, stackoverflow gibi web sitelerini kullanarak da bilgilerimizi pekiştirdik.

## 3- Modüller ve Açıklamaları

**Color** : Bir enum. Konsola renkli bir şekilde yazdırabilmek için böyle bir enum oluşturuldu.

**RandomEnumGenerator**: Proseslere rastgele renk atayabilmek için araştırmalarımızı yaparken Stackoverflow üzerinde bulduğumuz ve kullanmaya karar verdiğimiz bir sınıf. Rastgele şekilde enum üzerinden değer atamaya yardımcı oluyor.

**Proses:** Projemizin en temel sınıfı. Veri.txt dosyasından okunan satırları bu sınıf sayesinde proses nesnesi olarak gerçekliyoruz. Rastgele şekilde bir renk atanması da yine bu sınıfın kurucu fonksiyonunda gerçekleşiyor.

**Scheduler:** Bu sınıfımız prosesler arasından seçim yaparak onları Dispatcher'a iletiyor. Kurucu fonksiyonu parametre olarak bir Dispatcher nesnesi alıyor. Bu sayede direkt olarak Dispatcher'ın metodlarını kullanabilmemiz sağlanıyor.

**Dispatcher:** Scheduler tarafından gönderilen Proses'in CPU'ya atanması, Dispatcher'ın görevidir. Bu sınıfımızın kurucu fonksiyonunda ise Gerçek Zamanlı Kuyruk ile Kullanıcı Proses Kuyruklarını oluşturuyoruz. Bu sayede Dispatcher sınıfımız DispatcherControl() adlı fonksiyonunu kullanarak kendisine gönderilen Proses'in kendisine uygun kuyruğa eklenmesini sağlar. Bunun dışında DispatcherRun() fonksiyonu ilk olarak Gerçek Zamanlı Kuyruk'un boş olup olmadığının kontrolünü sağlar. Eğer boş değil ise Gerçek Zamanlı Kuyruk belirlenen şekilde çalıştırılır. Eğer boş ise aynı kontrol Kullanıcı Proses Kuyruk'u için kullanılır. Eğer boş değil ise Kullanıcı Proses

Kuyruk'u belirlenen şekilde çalıştırılır. Bu sırada Proses'lerin simüle edilebilmesi için ProcessBuilder kullanılır.

**RealTimeQueue:** Bu sınıfımız Gerçek Zamanlı Kuyruk'u gerçekleştirebilmek amacıyla oluşturulmuştur. Veri yapısı olarak LinkedList kullanılmıştır. Kurucu fonksiyonunda bu LinkedList oluşturulur. RealTimeQueueAdd(Proses proses) fonksiyonu ile LinkedList'e Proses eklenir. isEmpty() fonksiyonu boolean fonksiyondur ve LinkedList'in boş olup olmadığı kontrolunu sağlar. RealTimeQueueRun() fonksiyonu FCFS mantığıyla çalışır. Her ne kadar karmaşık döngüler ve if-else yapıları kullanılmış olsa da aslında yaptığı iş basittir. Bir proses askıya alınıp mı yürütüldüğünün kontrolu, Proses bitene kadar işlenmeye devam etmesi ve sonrasında da LinkedList'ten çıkartılarak sonlandırılması gibi işlevleri vardır.

**UserJobQueue:** Bu sınıfımız Kullanıcı Prosesleri Kuyruk'unu gerçekleştirmeyi amaçlamaktadır. Gerçek Zamanlı Kuyruk'tan daha karmaşık bir yapıya sahiptir. Öncelik sıralamasına göre üç adet seviyeden oluşur ve her proses 1 sn yürütüldükten sonra bir alt öncelikteki kuyruğa aktarılır. En son 3. Seviye kuyruğa inilir. Eğer üst kuyruklarda bir Proses kalmadıysa 3. Seviye kuyruk Round-Robin modunda çalışır. Bu sırada her an Proses'in gelişinden itibaren 20 saniye geçip geçmediği kontrol edilir. Eğer geçmiş ise Proses zaman aşımından dolayı sonlandırılır.

**Dosya:** Bu sınıfımız Veri.txt dosyamızın okunması ve Dispatcher ve Scheduler sınıflarının gerekli metodlarının çağrılması için kullanılır. İki farklı Oku() fonksiyonu bulunur. Biri parametresiz iken diğeri parametre olarak Veri.txt dosyasının dizin yolunu alır. Belirli döngüler sayesinde Kuyruklar tamamen boşalana kadar çalışmaya devam eder.

**Main:** Bu sınıf ise programımızın yürütüleceği sınıf. Program çalıştığı zaman belirli sorular ve if-else yapıları ile Dosya.Oku() fonksiyonlarından hangisinin çalışacağı belirleniyor. Mümkün olduğunca sade ve basit tutmaya çalıştığımız bir sınıf.

## 4- Olası İyileştirmeler

Özellikle askıya alınması gerekilen Proseslerde bu durumun yoklaması her saniyede bir yapılıyor. Ancak bu durumun kesme benzeri bir yapı ile denetlenmesi hem performans hem de simülenin gerçekliği için daha mantıklı olurdu. Bu projenin en büyük eksikliklerinden birisinin bu olduğunu düşünüyoruz. Bunun dışında bellek yönetimi konusunda bazı sıkıntılar olduğundan korkuyoruz, bu yüzden daha fazla Proses eklenmesi sonucunda bellek açısından sıkıntılar yaşanabilir. Bunun daha sıkı bir şekilde denetlenmesi gereklidir.