# http://upload.wikimedia.org/wikipedia/en/9/96/Sakarya_%C3%9Cniversitesi_(logo).jpg

# SAKARYA ÜNİVERSİTESİ

# BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ

# BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

# İşletim Sistemleri

# Proje / Tasarım

GitHup: https://github.com/acar2188/OS\_HW1

# **Grup-7**

# Fatih Acar-Y225012014

# Ufuk Akkaya-Y225012005

# Burak Gün-Y225012002

# Tarık Şener-Y225012012

Ahmet Kardeşseven-Y225012006

# 2.Öğretim B Grubu

Proje

**Proje Özeti**

Projede dört seviyeli öncelikli görevlendirici yapısı oluşturulmuştur. 1 adet İlk Gelen İlk Çalışır (FCFS) algoritması ile çalışan kuyruk ve üç seviyeli geri beslemeli round robin kuyruk bulunmaktadır.

Proje 2 kısımdan oluşmaktadır: main.java ve Process\_SW.java

**Main** kısımda giriş.txt okunarak bütün prosesler okunarak JobDispatchList’e eklenir. Sonrasında program çalışmaya başlar ve Thread.sleep((long)1000); fonksiyonu ile her döngüde 1 saniye geçmesi sağlanır. Her döngüde tick değiştkeni arttırılır ve program zamanı tick değişkeni ile takip eder. Her döngünün başında JobDispatchList kontrol edilir varış zamanı gelen prosesler ilgili kuyruğa eklenir. Main’in geri kalan kısmında görevlendirici algoritmalar bulunur.

**Process\_SW** Proses ile ilgili sınıf yapısı bulunmaktadır. Ayrıca ilgili proseslerin çalışma, askıya alınma ve sonlanması durumlarında ekrana basma fonksiyonları da bu kısımdadır.

Projede projeler ProcessBuilder ile oluşturulur.

ProcessBuilder PB;

ProcessBuilder sınıfının command fonksiyonu ile hangi programı ve hangi dosyayı açacağımızı belirttik.

PB.command("notepad.exe","cikis.txt");

ProcessBuilder sınıfının start fonksiyonu ile proses başlar. Ve start fonksiyonu java’nın kendi proses sınıfının nesnesi olarak döner. Biz de kendimiz için tanımladığımız HW\_PB prosesine atadık.

Process HW\_PB;

HW\_PB = PB.start();

Artık proses aşağıdaki fonksiyonlar ile bekleme, uyandırma ve sonlandırma işlemini yapabilmektedir.

HW\_PB.wait(1000);

HW\_PB.notify();

HW\_PB.destroy();

**Programın Çalıştırılması**

Proje dizinindeki Out klasöründe START.bat dosyası oluşturduk. İçerisinde çalıştırılabilir JAR dosyasını başlatan komut yazdık(java -jar "os\_hw1.jar"). Bu dosyaya çift tıklayarak açtığımızda cmd ekranı açlır ve JAR dosyasını çalıştıran komut çağrılır ve yazdığımız proje çalışmaya başlar. Video klasöründe 4x hızlandırılmış video bulunmaktadır. Ayrıca raporun sonunda ekran görüntüleri bulunmaktadır.

**Github Linki**

<https://github.com/acar2188/OS_HW1>

**Görevlendirici tarafından kullanılan yapıların tanımı ve tartışılması**

Yapı olarak bilgiler giriş.txt dosyasından aşağıdaki formatta okunmaktadır ve ilk olarak job list kuyruğuna eklenir. Ulaşma zamanı gelen görevler job listesinden önceliğine göre ilgili kuyruğa aktarılır Kuyruk isimleri: (FCFS,RQ0,RQ1,RQ2)

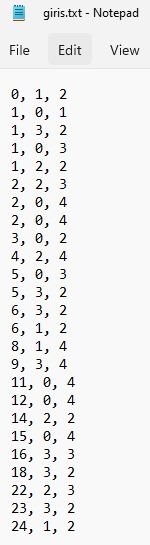


Figure 1:giriş.txt içeriği

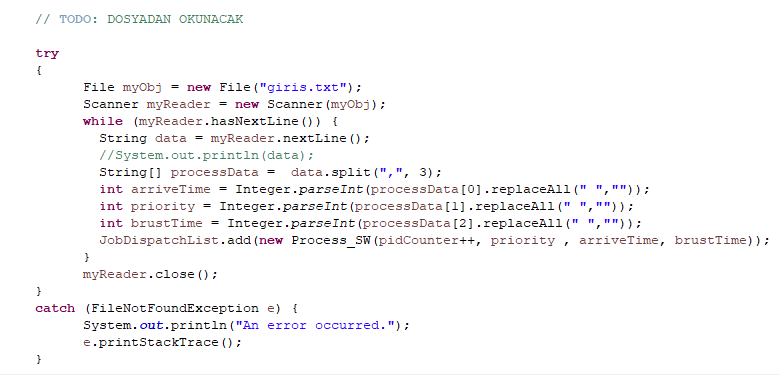
****

Figure 2:Dosyadan veri okuma

**Program yapısının ve bireysel modüllerin tanımı ve gerekçesi**

Program yapısının açıklamasını proje özeti kısmında yazdık.

* Main
  + Dosyadan okuma ve JobDispatchList’i doldurma
  + Tick zamana bağlı prosesleri kuyruklara ekleme
  + Görevlendirici yapısı
* Proses\_SW
  + Proses sınıfını barındırıyor
  + Proseslerin state bilgisine bağlı prosesleri başlatma, askıya alma ve sonlandırma yapıyor. Ve bu durumları ekrana basıyor.

**Görevlendiricinin tartışılması eksiklikler ve olası iyileştirmeler**

Burada bizim gördüğümüz en büyük eksik eğer FCFS’ye çok uzun sürecek bir proses gelirse bütün prosesler aksar. Bunun yerine FCFS kuyruğuna ortalama yanma süresini kapsayacak 5 saniye gibi bir değer belirleyip 5 saniyeyi aşan prosesler alltaki round robin kuyruğuna aktarılabilir. Böylece aşırı yüksek süreli prosesler sistemi kilitleme noktasına getirmez. Olası bir hata da tolere edilmiş olur.

**FCFS yüksek öncelikli sıralayıcının çalışması**

FCFS yüksek öncelikli bir proses varsa RQ0,RQ1 ve RQ2'den run durumunda olan proses varsa stop edilir, proses sonlandıysa prosesi listeden çıkartır, sonlanmadıysa öncelik düşürülür.

Detayı **main.java** içinde bulunmaktadır.

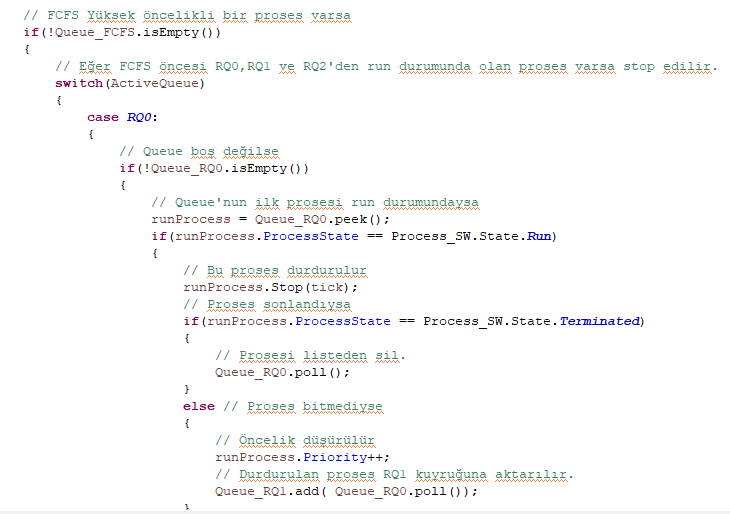
****

Figure 3:FCFS Algoritması

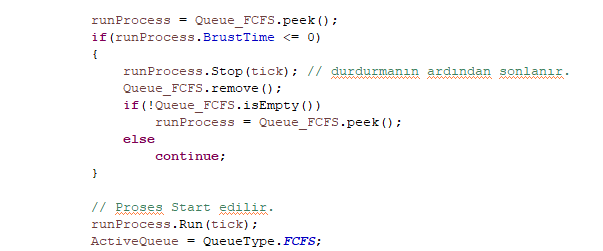


Figure 4:FCFS Process Çalıştırma

**Kullanıcı Geri Beslemeli sıralayıcının çalışması**

Geri beslemeli sıralayıcıda her proses 1 saniye çalışarak daha az öncelikli kuyruğa aktarılır. Son kuyrukta bütün prosesler bitene kadar döner.

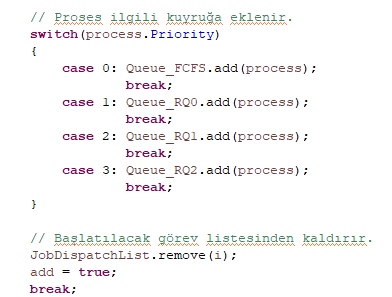
****

Figure 5:Her durum için tanımlamalar oluşturulmuştur

**Kullanıcı Geri Beslemeli sıralayıcının Round Robin modunda çalışması**

Yüksek öncelikli proses olmadığında çalışan süreçlerdir.

Detayı **main.java** içinde bulunmaktadır.

RQ0,RQ1,RQ2

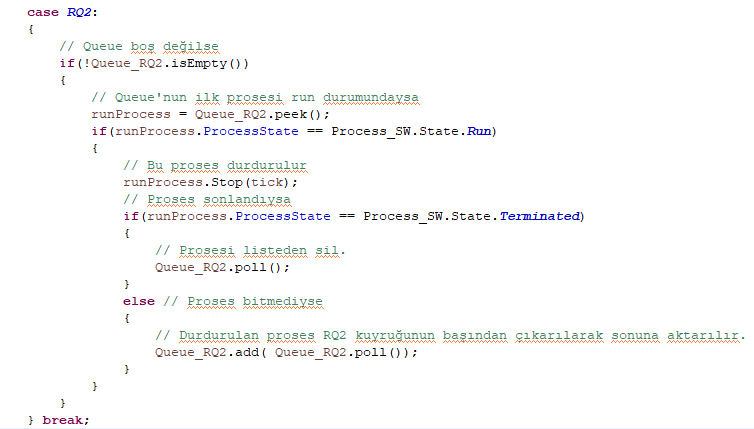
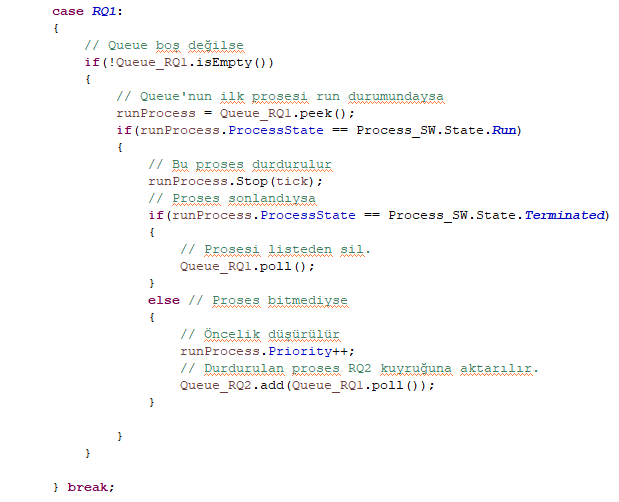
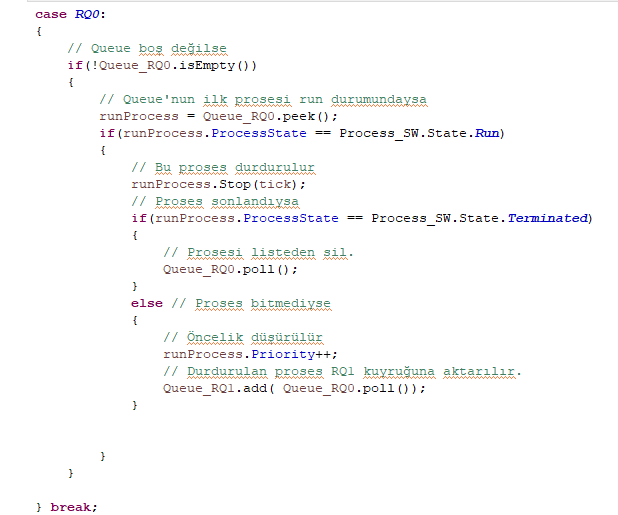
****

Figure 6:RQ0 Kuyruğu Figure 7:RQ1 Kuyruğu Figure 8:RQ2 Kuyruğu

**Karışık sıralayıcı çalışması**

Karışık sıralayıcı bütün görevlendiriciyi kapsamaktadır. Örneğin RQ0’da proses çalışırken FCFS’de proses gelmiş ise görevlendirici RQ0’daki prosesi durdurur ve FCFS’deki proses çalıştırılır. Aynı durum RQ0 proses geldiğinde RQ1 ve RQ2 için de yapılır. Onun dışında bütün kuyruklar ayrı bir şekilde çalışır. İlk önce FCFS kuyruğu sonra RQ0 sonra RQ1 sonra RQ2 kuyruğunda işler biter. En joblistte ve kuyruklarda proses kalmadığında program sonlanır.

**Kuyruk sınıfı**

Aşağıda sınıf içerisinde FCFS ve Round Robin kuyrukları bulunmaktadır.

Detayı **main.java** içinde bulunmaktadır.

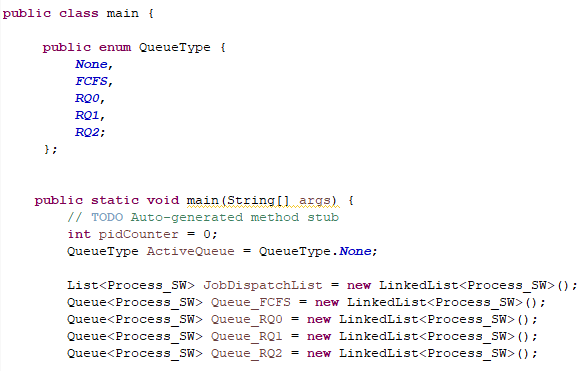
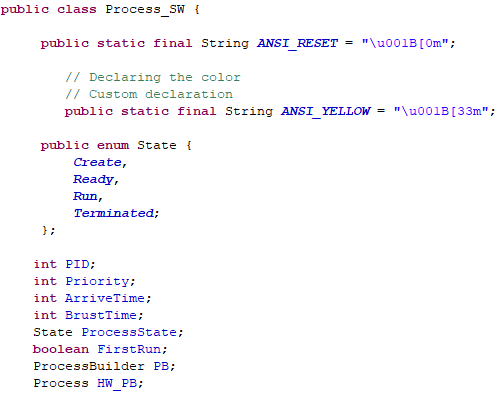
****

Figure 9:Proses kuyrukları

**Proses sınıf**

Aşağıda sınıf içerisinde process bilgileri bulunmaktadır.

Detayı **Process\_SW.java** içinde bulunmaktadır.



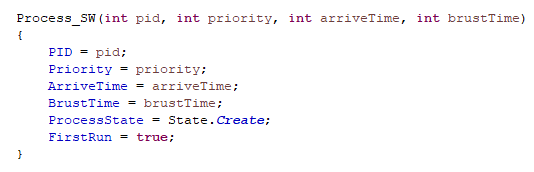
****

Figure 10:Proses Sınıfı

**Ana program - konsol uygulaması**

