Manisa Celal Bayar Üniversitesi Yazılım Mühendisliği Bölümü

YZM 2116- Veri Yapıları Dersi

Proje#1

"Bağlı Listeler, Yığın ve Kuyruklar"

Bahar 2018

Veriliş Tarihi: 05.03.2018
Son Teslim Tarihi: 26.03.2018
Proje Teslimi: cbuyzm2116@gmail.com

Projeyle İlgili Genel Kurallar:

1. Projeden tam puan alınabilmesi için, proje metninde yer alan, **Ödev#1** ve **Ödev#2**'nin yapılması gerekmektedir.
2. Proje, iki veya üç kişilik gruplar halinde yapılabilir.
3. Proje, dersi veren öğretim üyesine e-posta yolu ile (cbuyzm2116@gmail.com adresine) teslim edilecektir.
4. **Son teslim tarihinden sonra** teslim edilen projeler **değerlendirmeye alınmayacaktır.**
5. Farklı grupların teslim ettikleri projeler arasında kopya kontrolü yapılacak olup kopya teslim edilen tüm grupların, **dönem sonu proje notları** **"sıfır"** olarak değerlendirilecektir.
6. **Rapor içermeyen ödev teslimleri değerlendirmeye alınmayacaktır.**
7. Proje metninde yer alan, her iki ödev için **ayrı birer rapor** hazırlanmalıdır. Proje raporları, her iki programın, **açıklama satırları içeren kaynak kodları** ile birlikte, belirtilen son teslim tarihinde gönderilmelidir. Proje için, yazıcı çıktısı istenmemektedir.

   Rapor aşağıda belirtilen ilkeler doğrultusunda hazırlanmalıdır:

   - **Kapak:** Ödev numarası ve adı, Proje grubundaki öğrenci numaraları ve ad-soyadları ile proje teslim tarihi bilgilerini içermelidir.
   - **Rapor İçeriği:**
     i. Bölüm#1: Gerçekleştirilen Platform ve Dil (Java, C#, Eclipse, vs.) ve Sürüm Adı
     ii. Bölüm#2: Gerçekleştirimi Yapılan Problemin Kısa Tanımı (Ödev metninden özetlenebilir.)
     iii. Bölüm#3: Veri Yapısı Kataloğu (Kullanılan veri yapılarının, sınıfların ve metotların kısa açıklamaları)
     iv. Bölüm#4: Yazılım Geliştirme İçin Harcanan Süreler (Grup üyeleri için kişi ve saat bazında)
     v. Bölüm#5: Elde Edilen Örnek Sonuçlar (Bu bölüm ekran görüntüleri ve farklı durumlarda kodun çalıştığına ilişkin destekleyici kanıtlar içermelidir.)
     vi. Bölüm#6: Proje Gerçekleştirimde Yararlanılan Kaynaklar (Kod parçalarının alındığı ya da gerçekleştirimde yararlanılan başlıca kaynakların listelenmesi beklenmektedir.)

Ödev#1

"Kuyruk Gerçekleştirimi Tabanlı Otopark Benzetimi"

Ödev Metni:

a. Bir otoparkta, içlerinde sürücüleri bulunan *N* adet araba, otoparktan çıkarken bir kuyruk oluşturmaktadır. Her bir arabanın, otoparktan çıkış işlemi, 10 saniye ile 300 saniye arasında rastgele bir zaman almaktadır (Bir araba çıkmadan, diğer arabaya ilişkin çıkış işlemine başlanamamaktadır). Bunun için öncelikle, **ilk giren ilk çıkar (FIFO) yapısına dayalı bir kuyruk yapısı** tasarlayınız. Her bir arabaya numara vererek, her birine rastgele birer işlem süresi atayarak *N* elemanlı bir kuyruk oluşturunuz. Her bir arabanın işi biterek kuyruktan çıkarıldığında işlem tamamlanma süresini (işlem süresi dahil ne kadar süre kuyrukta kaldığını) liste halinde yazdırınız. Ayrıca, *N* araba için ortalama işlem tamamlanma süresini de hesaplayıp yazdırınız.

b. Aynı işlemi aynı değerleri kullanarak, **öncelikli kuyruk veri yapısı** için yineleyiniz. Öncelik kuyruğunda, işlem süresi en kısa olan araba ilk (öncelikli) olacak şekilde tasarlayınız (Kuyruk, elemanları küçükten büyüğe sıralı olarak tutmalıdır.) **Bu yapıda, hangi arabaların, FIFO kuyruğına kıyasla, daha az beklediğini işlem süreleri ve sıra numaralarıyla listeleyiniz. Ortalama işlem tamamlanma süresindeki kazancı (fark ve % cinsinden) hesaplayınız.**

Yukarıdaki seçeneklerde belirtilmiş gerçekleştirimleri C, C++, C#, Java ya da tercih edilen bir başka dil kullanarak yapınız. Arayüz ve sınıf tasarımına ilişkin herhangi bir kısıt bulunmamaktadır.

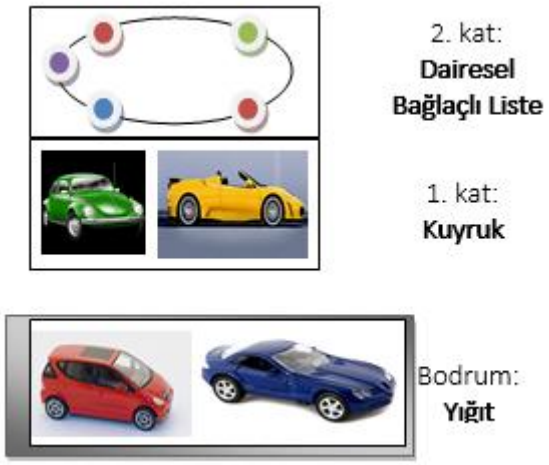
Önemli Not: Ödev#1 ve Ödev#2 birbirinden bağımsız iki ayrı projedir.

Ödev#2

"Yığın, Dairesel Bağlı Liste ve Kuyruk Tabanlı Otopark Benzetimi"

Ödev Metni:

Bir otoparkta bulunan arabaların iş çıkışında durumlarını gözlemlemek için bir benzetim programı geliştirilmesi amaçlanmaktadır. Otomobiller, otoparkın **zemin katında (1. Kat) bir kuyruk**, **bodrum (yer altı katında) bir yığıt** ve **ikinci katında ise bir dairesel bağlı liste** düzeninde yerleştirilerek çıkabilmektedir. Otoparkın genel yerleşim planı şekilde özetlenmiştir:



Arabalar otoparktan birer birer ve sadece birinci kattan çıkabilmektedir. Ardından, %50 ihtimalle bodrum, %50 ihtimalle ise ikinci kattan gelen bir taşıt, birinci kattaki kuyruğa eklenmektedir. Sadece arabaların, bağlı liste düzeninde yerleştirildiği kattan inecek araba, "Josephus problemi" benzeri *n* adet araba atlanarak seçilmektedir [Josephus problemi için https://en.wikipedia.org/wiki/Josephus_problem] ve bağlı listelerde kalınan yer tutulmaktadır.

a) **Tüm veri yapılarına 15'er adet araba yerleştiriniz ve listeyi ekrana yazdırınız. Arabaların her birinin farklı renklerde olduğunu düşünüp Renkleri (Kırmızı, Yeşil, Mavi,… gibi) "text" olarak temsil edebilirsiniz.**
b) **Tüm arabalar bitene kadar yukarıda anlatılan işlemi tekrarlayınız ve her turda, katlarda kalanların, silinenlerin renklerini ekrana yazdırınız. Son kalan arabanın rengini belirtiniz.**
c) **Kullandığınız bilgisayarın 5 saniyede ortalama kaç adet otopark problemi çözebildiğini hesaplatınız.**

Yukarıdaki seçeneklerde belirtilmiş gerçekleştirimleri C, C++, C#, Java ya da tercih edilen bir başka dil kullanarak yapınız. Arayüz ve sınıf tasarımına ilişkin herhangi bir kısıt bulunmamaktadır.

Önemli Not: Ödev#1 ve Ödev#2 birbirinden bağımsız iki ayrı projedir.