2018-19 Bahar

25.03.2019

BİL 2002 Nesneye Yönelik Programlama

Proje 1 Teslim Tarihi: 8 Nisan 2019 Pazartesi 21:30 (GitHub üzerinden)

Proje Konuları (2 veya 3 kişilik gruplar halinde)

Önemli Kural ve Koşullar:

- Gruplar, aşağıda konuları verilmiş olan üç farklı projeden herhangi **bir tanesini** seçebilir.
- Projelere ilişkin tüm kaynak kodlar, dosyalar, dizinler, açıklama/referans metinleri GitHub üzerinden yüklenecektir. Her gruptaki öğrencilerin farklı yüklemeleri değerlendirileceği için, projenizde *sadece tek bir öğrencinin* aktivitelerinin bulunması durumunda diğer grup üyeleri **not alamayacaktır**.
- Proje teslimi için ek süre kesinlikle verilmeyecektir. Herhangi bir nedenle zamanında iletilmeyen projeler, hiçbir mazeret kabul edilmeden 0 (sıfır) olarak notlandırılacaktır.
- Farklı proje ekiplerinin birbirinden ya da üçüncü şahıslardan kopya / intihal durumu belirlendiğinde, ilgili tüm proje ekiplerindeki öğrenciler projeden 0 (sıfır) alarak ilgili üniversite yönetmeliği çerçevesinde değerlendirileceklerdir.
- Projenizde C#, Java veya C++ programlama dillerinden herhangi bir tanesini kullanabilirsiniz. Projelerinizin derlenebilmesi ve çalıştırılabilmesi için gereken ekstra bilgilerin belirtilmesi proje gruplarının sorumluluğundadır. Farklı kurulum gerektiren projeler değerlendirilmeyecektir.
- Projelerinize **not verilip değerlendirilirken aşağıdaki teknik gereksinim ve kıstaslara öncelikle dikkat edilecektir:**
    - Projede geliştireceğiniz programınızda, ilgili tüm hesaplar, işlemler, değişkenler, vb. sizin ayrıca tanımlayacağınız sınıflar (class) içerisinde yer almalıdır. Bu gibi işlemler ve değişkenler, ana program içerisinde olmayacaktır. Ana programdan ise, bu sınıftan nesne / nesneler yaratılarak programın çalışması sağlanmalı ve gerekli veri girişleri ve hesaplama, ekrana yazdırma, vb. işlemlerin hepsinin bu nesne / nesneler üzerinden çağrılarak yapılması gerekmektedir.
    - Projenizde **gerekli ve uygun olan** yerlerde;
        - Ana sınıflardan (base classes) türetilmiş alt sınıflar (derived classes), interface'ler ve kalıtım (inheritance) özellikleri kullanılmalıdır.
        - Sarmalama (encapsulation) kullanılarak bazı metot (method) ve değişkenlerin farklı seviyelerde erişim kısıtlama olanakları (access modifiers) kullanılmalıdır.
        - Hiçbir değişkene (public de olsa) doğrudan erişim **yapılmamalı**, bunun yerine **property** (get / set) **kullanılmalıdır**.
        - Bir ya da birkaç method için çok biçimlilik (polymorphism) kullanılmalıdır (Compile time polymorphism; Overloading ve/veya Runtime polymorphism; Overriding).

KONULAR

1. Sözcük Tahmini:

Bu projede sizden, bilgisayarla oynayabileceğiniz bir sözcük tahmin oyunu yazmanız beklenmektedir.

Program, kullanıcıdan ismini alarak başlayacak; sonrasında "Yeni Oyun" seçeneği girildiğinde tahmin edilecek sözcüğün harf sayısını ve oyunun zorluk derecesini soracaktır.

Zorluk derecesi, sözcüğün basit, türemiş veya birleşik sözcük olması ile ilgilidir.

Örneğin: uçak(basit kelime- kolay), sulama(türemiş kelime- orta) , biçerdöver(birleşik kelime-zor)

- Harf sayısı 2 ve 12 arasında olabilir
- Kendi başına anlamı olmayan (edat,bağlaç vs.) sözcükler kullanılamaz. (ve,bile, gibi..)
- Aynı dile ait sözcükler kullanılmalı. (Oyun Türkçe ise Türkçe, İngilizce ise İngilizce)
- Oyunda en az 3 zorluk derecesi olmalıdır, (Kolay, Orta ve Zor)
- Oyun için bir sözlük oluşturmanız gerekmektedir. Bu sözlükte en az 50'şer adet basit, bileşik ve türemiş sözcük yer almalıdır. (internette TDK vb. sayfalardan yararlanabilirsiniz.)
- Oyun, oluşturulan sözlük kullanılarak programca rastgele belirlenen sözcüklerden 10 adet seçilerek sırasıyla kullanıcıya sorularak oynanır.
- Kullanıcıya tüm 10 sözcüğü tahmin etmesi için verilecek süre belirlenir. (4, 5 veya 6 dk. uygun olabilir.)
- Sözcükler küçük harf sayısından büyük harf sayısına doğru ilerleyen sırayla kullanıcıya sorulur.
    - Sözcük 2 harfli, 2. Sözcük 3 harfli, …., 10. Sözcük 11 harfli gibi)
- Oyun, kullanıcının belirlenen süreden önce tüm sözcükleri tahmin etmesi veya süresinin dolması ile sonlanır.
- Her bir tahminden sonra program (AI), sonraki tahminler için kullanıcıya iki farklı ipucu verecektir:
    - Yapılan tahmindeki harflerden hangilerinin sözcük içerisinde varolduğu
    - Yapılan tahmindeki harflerden hangilerinin sözcük içerisinde de aynı sırada yer aldığı
    - Örneğin sözcük "Maket" olduğunda:
    - Yapılan tahmin "Çomar" olduğunda verilmesi gereken mesaj şöyledir:
        - Sözcükte yer alan harfler: 'YOK'
        - Eşleşme sağlanan harfler: YOK
    - Yapılan tahmin "Saçak" olduğunda verilmesi gereken mesaj şöyledir:
        - Sözcükte yer alan harfler: 'k'
        - Eşleşme sağlanan harfler: YOK
    - Yapılan tahmin "Melek" olduğunda verilmesi gereken mesaj şöyledir:
        - Sözcükte yer alan harfler: 'm', 'k'
        - Eşleşme sağlanan harfler: 'm'
    - Yapılan tahmin "Maket" olduğunda verilmesi gereken mesaj şöyledir:
        - Tebrikler, bildiniz!

- Kullanıcıdan tahmin alınması esnasında geçersiz sözcük girişleri kontrol edilmelidir. Harf yerine rakam girişi, soru için belirlenmiş harf sayısından eksik veya fazla harf içeren sözcük girişi, noktalama işareti yazımı ya da oyun diline ait olmayan karakter girişi (Türkçe ise w,q,x veya İngilizce oynanıyorsa ü,ğ,ç gibi) exception handling yapılarak uyarı mesajı görüntülemeli ve yeniden giriş istenilerek oyuna devam edilmelidir.
- Oyun sonunda her bir sözcüğün kaç tahminde bilindiğini, Ortalama tahmin süresini(ipucunun gösterilmesi ve yeni tahmin yapılması arasında geçen sürenin saniye cinsinden miktarı), Toplamda geçen süre ve kullanılan harf sayısı, en çok kullanılan harfler vs gibi birtakım istatistiksel bilgiler gösterilmelidir.
- Her oyun sonlandıktan sonra oyuncuya tekrar oynayıp isteyip istemediği sorulacaktır.
- Eğer oyuncu isterse oyunu aynı ayarlarla tekrar başlatabilmelidir.

- Oyunun bitiminde, oyuncu tekrar oynamak istemediğinde çıktığı ana ekranda kendisine ait genel performans istatistiklerinin verildiği bir Log (Seyir Defteri) seçeneği bulunmalıdır. Burada kullanıcının, her basamak ve zorluk derecesi bazında oynadığı oyunları kazanma oranı (kaç oyun kazandığı veya kaybettiği), oyun başına yaptığı ortalama tahmin sayısı, bunların kaç tahminde sonlandığı bilgileri özet bir şekilde dökülmelidir. Böylece oyuncu, kelime dağarcığının ne kadar geniş olduğunu anlayabilecek, dünyaya url.siyle belgesiyle sunabilecektir.
- Projede en azından **Sozcuk**, **Tur**, **Oyun**, **AI**, **Oyuncu**, ve **Kaydedici** sınıfları olması beklenmektedir.
- AI sınıfı sözcük seçme, Oyuncunun sınıfının örneklerinden gelecek tahminlere cevap verme gibi görevleri yerine getirecektir.
- Oyuncu sınıfı ise kullanıcıdan alacağı girdi ışığında AI ile iletişime geçerek kullanıcıdan aldığı bilgileri AI sınıfına/nesnesine göndermek ve gelen sonuçları program kullanıcısına bildirmekle görevlidir.
- Bu iki sınıfın örneklerini içerecek olan Oyun sınıfı ise oyunu başlatma ve bitirme görevlerini yerine getirecektir.
- Kaydedici sınıfı ise log ve istatistiklerden sorumlu olacaktır.
- Projenizde oyun ayarlarını kod dışarısında değiştirmenizi ve saklamanızı sağlayacak bir konfigürasyon dosyası kullanabilirsiniz.

Programınızın giriş noktası olan Program sınıfındaki Main metodu ise **yalnızca** Oyun sınıfının bir örneğine sahip olacaktır.

2. Otel Rezervasyon Sistemi:

İstenen rezervasyon sistemi, kullanıcıların bir otelden, otel odası rezervasyonu gerçekleştirebilecekleri bir yazılım uygulamasıdır. Sistem kullanıcıların, sistem üzerinden uygun koşullara sahip otel odalarının listelenmesi (Tek yataklı, Çift yataklı ve İkiz yataklı; Deniz manzaralı ve Havuz manzaralı vb..) ve istenen tarihler arasında boş odaların arasından rezervasyon yapmasını sağlayacaktır. Uygulamanızı konsol üzerinden veya form üzerinden gerçekleştirebilirsiniz. Sistem üzerinden rezervasyon gerçekleştiği zaman ilgili odanın rezerve olarak işaretlenerek devamında gelecek işlemlerde buna göre ele alınması gerekmektedir. Ayrıca istendiği taktirde gerçekleştirilmiş rezervasyonlar sistem üzerinden iptal edilebilecektir.

Uygulamanızda minimum 3 sınıf bulunmalıdır: *Rezervasyon* sınıfı, *Otel* sınıfı, *Oda* sınıfı. Bu sınıflara ek olarak başka sınıflar da tanımlayarak tasarımınızı detaylandırabilirsiniz.

Uygulama Gereksinimleri:

Otel sınıfı

- Oda koleksiyonlarının tutulması,
- *Rezervasyon* sınıf örneklerinin çağıracağı oda müsaitlik sorgu metotları,
- *Rezervasyon* sınıf örneklerinin çağıracağı rezervasyonu gerçekleştirme metodu
- *Rezervasyon* sınıf örneklerinin çağıracağı rezervasyonu iptal metodu,
- Oteldeki odalarla ilgili doluluk oranı, belirli tipteki odalardaki doluluk oranı, belirli bir tarih için doluluk oranını hesaplayacak metotları sağlar.

Oda sınıfı

- Oda özelliklerinin tutulması,
- Odanın tarihlere göre müsaitlik durum bilgisinin tutulması,
- *Otel* sınıfı örnekleri tarafından çağrılacak rezervasyonu gerçekleştirme metodu,
- *Otel* sınıfı örnekleri tarafından çağrılacak rezervasyonu iptal metodunu sağlar.

Rezervasyon sınıfı

- Sistem üzerinden oda sorgularını gerçekleştirmek,
- Rezervasyon isteğini otele iletmek,
- Rezervasyon iptalini otele iletmek,
- Otelin doluluk oranları ile ilgili istatistiki bilgileri otelden istemek işlemlerini sağlar.

3. <u>Dosyadaki Hakikat:</u>

Bir dosyadan hatalı sayı girişlerini saptayıp ayıklayan bir C# uygulaması hazırlayınız. Bu program hatalı değerleri filtrelemeli ve hatasız sayıları (tamsayı veya ondalıklı) sıralı bir şekilde (sorted) çıktı olarak vermelidir.

Uyulması gereken kurallar aşağıda belirtilmiştir:

- Okunacak dosya komut satırı argümanı olarak alınmalıdır. Ancak, komut satırı argümanı girilmemişse, kullanıcıya okunacak dosyanın adı sorulmalıdır (geçerli bir dosya adı girene kadar).
- Dosyadaki her satırın sadece tek bir sayı içermesi beklenmektedir, ancak bazıları hatalıdır. Her satırı programdan okutunuz ve geçerli sayı değerlerini ortaya çıkartınız (yani hatasız olanları elekten geçiriniz).
- *Hatalı girişlere örnekler*:
  - A // `karakter girdisi`
  - SdgA // `string girdisi`
  - 1R5 // `içinde harf bulunan karakter girdisi`
  - 56,1 // `virgül ile ayrılmış ondalıklı sayı`
  - !
  - 9?4
  - 9-0
  - ***
- *Hatasız ve sayı olarak değerlendirilmesi gereken girişlere örnekler*:
  - 405
  - 43.7
  - -78
  - -3.14
- Programınız hataların üstesinden gelebilmeli (exception handling ile) ve dosya sonuna gelene kadar geçerli sayıları okuyabilmelidir. Programınızın içerisinde Sayi abstract sınıfından türetilmiş TamSayi ve NoktaliSayi sınıfları mevcut olmalıdır. Sayilariniz bu sınıflar vasıtası ile tutulmalıdr.

- Dosyada kaç tane sayı olduğu önceden bilinmemektedir. Programınız dosya sonunu tespit etmelidir.
- IntSort isimli ayrı bir sınıf oluşturunuz. Bu sınıfın kurucu metodu dosyadan okunmuş olan sıralanmamış Sayi dizisini parametre olarak almalı, bu sayıları sıralayarak sıralanmış halini sınıf attribute'ü olarak kaydetmelidir. IntSort sınıfının ayrıca sıralanmış diziyi döndürdüğü bir metodu daha olmalıdır. Herhangi bir sıralama algoritmasını kullanabilirsiniz.
- Programınız geçerli sayıları sıraladığı gibi, dosyadaki toplam satır sayısını belirtmeli, kaçının geçerli sayı içerdiğine dair sayısal ve yüzdesel bilgi vermelidir. Örneğin:
    - Toplam 194 satır
    - 121 geçerli sayı değeri bulundu (%62.37)
    - 73 geçersiz satır (%37.63).
- Programınız boyunca karşılaşılan tüm exceptionlar bir log dosyasında tutulmalıdır. Formatını öyle bir şekilde ayarlayınız ki aşağıdaki son maddedeki şartlar karşılansın.
- Programınız bu log dosyası adı (geçerli bir dosya yoluyla) ve ek bir opsiyon (örneğin "-L" şeklinde) içerecek şekilde iki komut satırı argümanıyla çağrıldığında, hangi tarihte (gün bazında) ve toplamda kaç adet hatanın kaydedilmiş olduğu ekranda listelenebilmelidir.

Başarılar, iyi çalışmalar 😊