KARADENİZ TEKNİK ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ
NESNE YONELIMLI PROGRAMLAMA

ÖDEV RAPORU

Öğrenciler: Ertuğrul Bilgiç 402545, Hüdahan Altun 394753, Irmak Sılay Kara 402497

Ders Adı: Nesne Yönelimli Programlama

Ders Sorumlusu: Dr.Öğr.Üyesi Beste Üstübioğlu

Grup: G15

Ödev Verilme Tarihi: 19/10/2021 13.38

Ödev Teslim Tarihi: 01/01/2022 01.00

Ödev Savunma Tarihi: 07/01/2022 17.00

Konu:

2021-2022 Eğitim Öğretim Yılı Güz Dönemi Nesne Yönelimli Programlama Dersi Dönem Ödevi içeriğine dair rapor

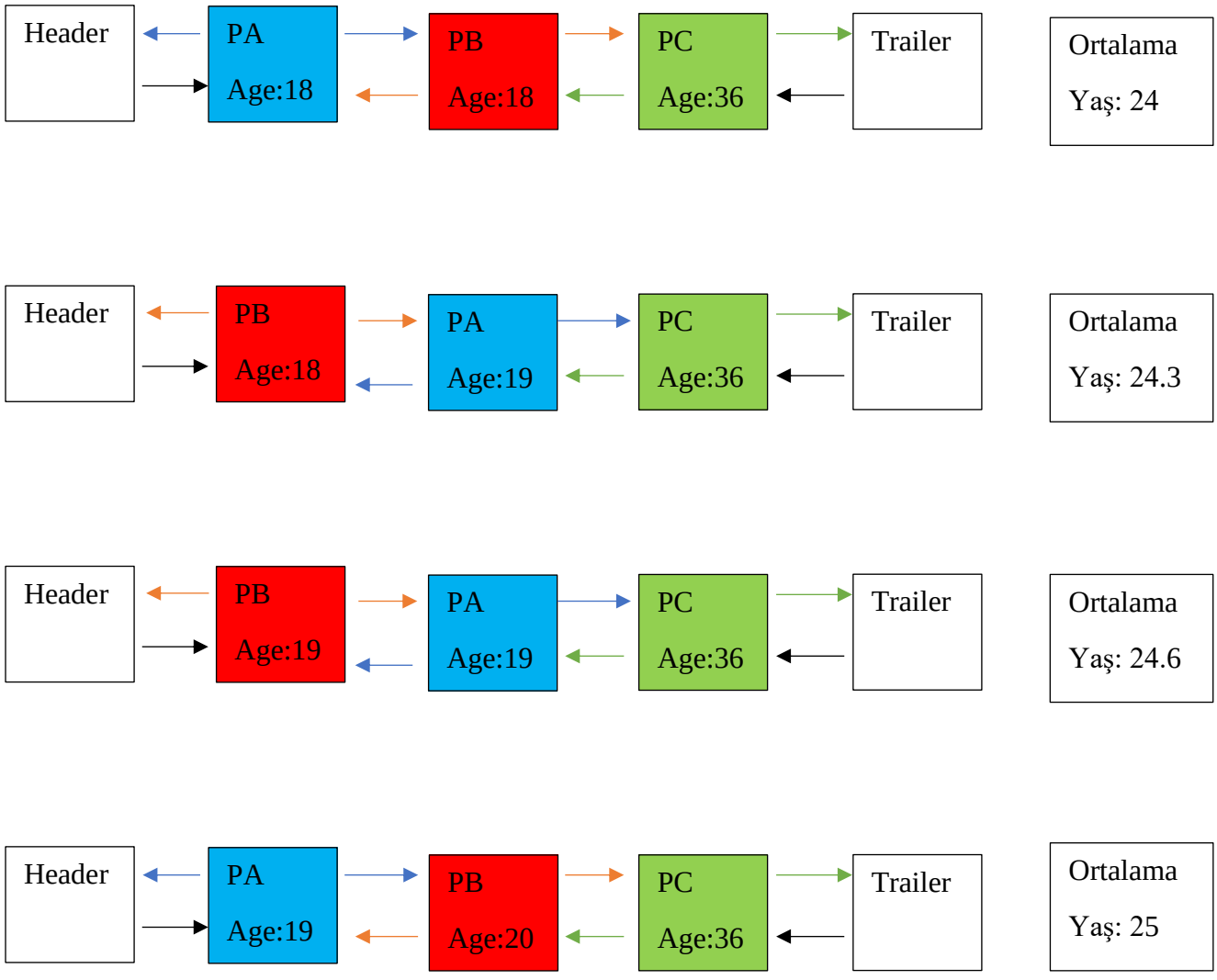
Rapor:

Ödevde istenilen şartlar bizlere “Nesne Yönelimli Programlama Dersi Ödevi.docx” belgesinde iletilmiş olup bizden istenilen ile bizim istenileni nasıl yazdığımız aşağıda belirtilmiştir.

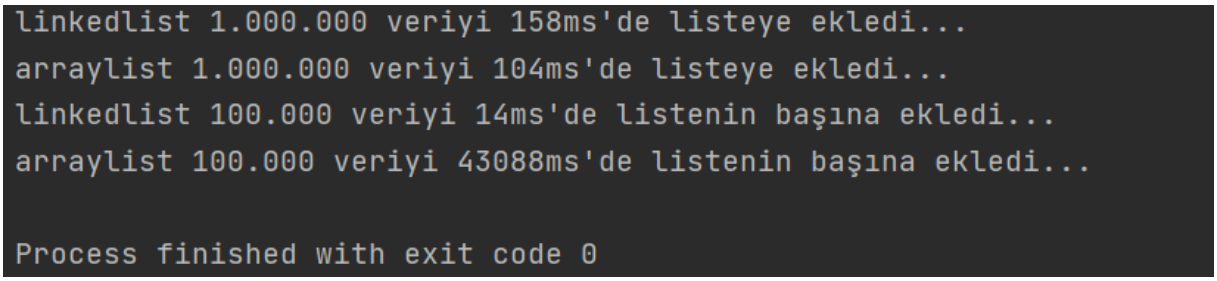
Ödevin ilk maddesi bizden varsayılan 18 olmak üzere üzerinde oynama yapılabilir sayıda takımdan oluşan ligimizin takım adlarının “takimlar.txt” dosyasından okunup kısaltmalarının oluşturulma şeklini belirtmektedir. 2353 satırlık projemizde bu işlem “League” sınıfında gerçekleştirilmektedir. Öncelikle txt’deki veriler iki adet vektöre kaydedilir ve bunlardan takım adlarını tutan vektör kısaltma oluşturucu bir fonksiyona sokulup kısaltma adlarını tutan bir vektör oluşturulur.

Ardından “for” içerisinde ilk girişte alınan takım sayısına göre takımlar oluşturulur. Kısaltma oluşturma sistemimiz ise takımın ismine göre değişmektedir. Eğer takım içinde “spor” kelimesi geçmekte ise kısaltma sonuna “-s” eklemektedir ve çakışmaları engellemek için içinde “spor” geçen takımlar ekstradan 3. harfi de almaktadır. Takım adları menüde yer alan takımları listele tuşuna basılınca ayrıntıları ile gösterilmektedir. Takımın kısaltması girilerek bu takımın değerleri ayarlanabilmektedir bu takımın değerlerinin ayarlanmasına girildiğinde de bir oyuncunun adı soyadı yazılarak bulunmaktadır.

Ödevin 2. Maddesi için Futbolcular “Team” sınıfı constructor’ı içinde tıpkı takımların oluşturulması gibi txt’de yer alan isimler ve soyisimler bir vektöre eklenerek rastgele bir indexteki isim ve soyisim futbolcuya atanmaktadır. Futbolcunun diğer statları da yine dokümantasyonda belirtilen değerler içinde rastgele olarak belirlenmiştir oluşturulan futbolcular kendi yazdığımız Doubly Linked List yapısında tutularak takım için gereken minimum yaş ve performans şartları kolaylıkla ve yüksek verimlilikle sağlanabilmektedir.



Şekil 1: Projede yaş ortalaması nasıl çalışmakta

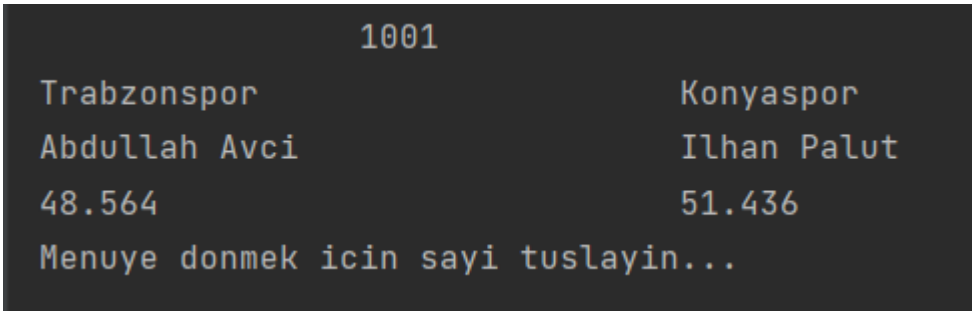


Şekil 2: Neden Array yerine Linked List

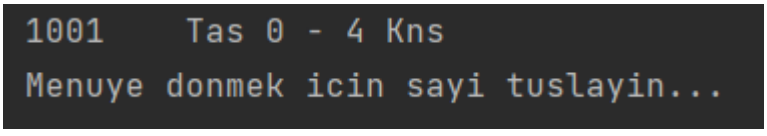
Oyuncuların oluşturulması sırasında Linked List yerine Array kullanılsaydı NFL gibi çok takımın yer aldığı ve çok fazla oyuncu bulunduran bir ligin load süresi uzun sürerdi.

Futbolcunun adları veya herhangi diğer bilgileri "Takımları Listeleme> Takım Kısaltması> Oyuncu adı soyadı" şeklinde yapılan tuşlamalarla ulaşılabilmektedir ve değiştirilebilme olanakları sağlanmıştır fakat "call by reference" yapıldığı halde değiştirme yapılamamaktadır. Sebepleri araştırma süresinin yetmemesi sebebi bulunamamıştır.

Üçüncü madde için takımların çakışmayan maçları "League" sınıfı içinde programın başlatılması ardından txt'den veriler çekilmiş ve eğer 18'den fazla takımla lig oynatılıyorsa fazladan eklenen takımlar eklendikten sonra lig fikstürü otomatik olarak oluşturulur. Fikstür tamamen gösterilmeyip id'si girilen maç eğer oynandıysa ayrı oynanmadıysa ayrı olarak gösterilmiştir. Takımlar maçları tek tek oynayıp maçlar arası takımlara değişiklik yapılması sağlanmıştır.



Şekil 3: Maç oynanmadan önce id ile maç bilgisi sorgulama



Şekil 4: Maç oynandıktan sonra id ile maç bilgisi sorgulama

Dördüncü maddede oyuncuların tek tek karşılaştırılmasının Dokümantasyon analizinde fark edilmemesi sonucu minimal Şekil 3'deki gibi iki takımın isim ve teknik direktörü ve kazanma olasılığı verilmiştir. Aynı maç oynandıktan sonra incelendiğinde ise Şekil 4' de yer alan görüntü ortaya çıkmaktadır.

Beşinci Madde Futbolcu bilgilerinin değişmesinde olduğu gibi takımı oluşturan diğer verilerin değişmesinde de rastlanan “call by reference” yapıldığı halde değişmemesi sonucu maç sonucu lig tablosunda gözlemlenememektedir.

Son madde ise “Match” sınıfındaki “playMatch” fonksiyonu ile yapılmaktadır. Bu fonksiyon sınıf özelliği olarak yerleştirilen iki takım arasında dokümantasyonda istenilen şekilde maç yaptırmaktadır. Takımların ortalamaları “DoublyLinkedListForPlayer” sınıfındaki “totalAP”, “totalMP”, “totalDP” şeklindeki fonksiyonlar ile alınıp “Team” sınıfı içindeki “calcMeanAP”, ”calcMeanMP”, ”calcMeanDP” ve buradan elde ettiği sayılar ile “calcMeanPower” fonksiyonu ile hesaplanmaktadır.

Eğer AP, MP, DP değerleri 60’dan küçükse Şekil 1’e benzer bir mekanikle takım güç ortalamaları 60 seviyesine getirilmektedir. 0-6 arası rastgele skorlar üreterek başlayan maç bu ortalamalar ile ilgili işlemeler yapılarak kazananı belirlemekte son olarak kazanan takımın genel ortalamasına göre yine rastgele bir şekilde ya kendi skoruna +1 ya da karşı takım skoruna -1 eklenerek maç sonucu takımların ilgili fonksiyonlarına iletilir ve skorlar ayrıca “Match” sınıfında tutulur.

Kod genel olarak incelendiğinde ise olabilecek her noktada açıklamalar yapılmış olup main metodu 2353 satırlık projenin sadece 6 satırını kaplamaktadır. Bu da modülerliğin sonuna kadar kullanıldığının kanıtı olup ayrıca yazmış olduğumuz “Texts” sınıfı sayesinde proje içerisinde yer alan stringler tek bir sınıf içinde tutularak olası yazım hatası durumunda tek noktadan değiştirilmesini sağlamaktadır. Bunun yanında bu sınıf sayesinde tekrarlı yazım olmamış ve kod akıcılığı sağlamıştır. Kod yazımında “camelStyle” kullanılmıştır.

OOP prensiplerinden comprehension; bir ligin takım ve maçlardan, bir maçın takımlardan, takımların oyuncu listelerinden ve listelerin oyunculardan oluşmasından kaynaklı proje genelinde görülmektedir. Inheritance ise oyuncu listleleme tekniğinde sıkıntı yarattığı için kullanılmamış olup bu yanında polymorphism’inde görülmemesine sebep olmuştur. Ancak bir sınıfa ait non-const verilerin büyük bir kısmı private tanımlanarak encapsulation kullanımı sağlanmıştır.