SAKARYA
ÜNİVERSİTESİ

11 MAYIS 2024

PROGRAMLAMA DİLLERİNİN PRENSİPLERİ

2. ÖDEV

SUDE ÖZKANOĞLU
G201210034
sude.ozkanoglu@ogr.sakarya.edu.tr

RAPOR

Ödev çalışmasında ilk olarak gönderilen ödev raporunu inceleyerek çalışmalara başladım. Ödev isterlerini doğru bir şekilde tamamlayabilmek adına "Kalıtım ve Soyut Sınıf Benzetimi" ve "Nesne Yönelimli Benzetim" videolarını izleyerek bir temel yaparak devam ettim. İlk olarak bu videolar içerisindeki yazılan kodları tekrar yazarak kalıtım ve benzetim konusundaki bilgilerimi tazeleyip eksiklerimi giderdim. Ardından ödev için gereken minimum yapıların dosyalarını klasör hiyerarşisi altında oluşturarak içlerini doldurmaya başladım. Dosya okuma olmadan ödevin herhangi bir ilerleme kaydedemeyeceğini bildiğimden ilk olarak o kısmı bitirmeye odaklandım.

Dosya okuma ile ilgili header dosyamı oluşturduktan sonra içerisinde gerekli methotları oluşturdum. Bu metotlar toplam sayı sayısını hesaplayan, satır sayısını hesaplayan ve 1 ile 99 arasındaki sayıları oluşturduğum dizi içerisine atan metotlardı. Bu metotların doğru bir şekilde sonuç verebilmesi için metotlar arasındaki parametre ilişkilerini tamamladım ve dönüş değerlerini doğru bir şekilde gerçekleştirdim. Oluşturduğum alanları yok edebilmek adına yıkıcı metodumu da ekledim.

Kalıtım yapısını doğru bir şekilde sağlayabilmek için ödev dosyasında da belirtildiği gibi *gorunum* fonksiyonunu *Canlı.h* içerisinde yaratarak çalışmalara başladım. Buradaki işlemlerin doğru bir şekilde tanımlanması noktası en zorlandığım kısım oldu. Özellikle alt sınıflarda hangi yapılara ihtiyaç olabileceğini düşünerek bu sınıfın doğru bir şekilde yaratılması kısmı en çok zorlayan ve ödevin zaman harcandığı kısım oldu. *Canlı.h* dosyasını başarılı bir şekilde oluşturduktan sonra metot gövdelerini doldurdum. *Canlı.c* içerisindeki metotların kullanılabilmesi için Bitki, Böcek, Sinek ve Pire yapılarını oluşturmaya başladım. Kalıtım olarak en kolay basamak olan Bitki sınıfını gördüğüm için ilk olarak onun yapılarını oluşturdum. Burada yapıcı metot içerisinde hangi parametrelerin dışarıdan geleceği ve bunları nasıl kontrol edeceğim kısmı diğer bir zorlayıcı kısım oldu. Alt sınıfların nelere sahip olacağı ve bu yapıları nasıl kontrol edeceğini ayarlayabilmek benim için en zorlayıcı ayrıca en çok bilgi birikim kazandığım kısımdı.

Bitki sınıfını başarılı bir şekilde oluşturup Canlı ile bağlantısını sağladıktan sonra Böcek sınıfını oluşturmak çok zor olmadı. Genel olarak Bitki ile aynı yapıyı takip ettiğinden bu kısmı zorlanmadan atlattım. Kalıtımın doğru bir şekilde sağlanabilmesi için Sinek ve Pire sınıflarının

oluşturulması diğer alt sınıflara göre daha zorlayıcıydı. Özellikle yapıcı metot içerisinde gelen parametrelerin doğru bir şekilde en üst sınıfa kadar ulaştırılması gerektiği kurgusu buradaki farklı bir kazanımdı. Bu kısmı başarılı bir şekilde tamamladıktan sonra tüm alt sınıflar ve ana sınıf arasındaki yıkıcı metot bağlantılarını da sağlayarak Habitat sınıfının yapısını kurgulamaya başladım.

Habitat sınıfının yapısını kurgulamak için önce bu sınıf içerisinde nelere sahip olmam ve neleri kontrol etmem gerektiği üzerine kafa yordum. Uzun değişiklikler ve tıkanıklıklar sonrası canlıların oluşturulması, anlık görünüm, birbirini yeme ve yıkıcı fonksiyonların oluşturulmasına karar verdim. İlk olarak canlıları doğru bir şekilde oluşturabilmek için ödev isterleri içerisinde verilen sayı aralıklarını kontrol eden ve aynı zamanda doğru sayı aralıkları için doğru objeleri oluşturan döngüyü yazdım. Döngü içerisinde oluşan elemanlara daha sonra birbirini yeme fonksiyonu içerisinde ihtiyacım olacağından, oluşan her bir elemanın adres değerini kaybetmemek adına her bir canlının tipinde dizi oluşturdum. Oluşturduğum bu elemanları dizi içerisinde tutarak doğru parametreleri ve index değerlerini yapıcı metotlar vasıtasıyla gönderdim. İlk durumun ekrana doğru bir şekilde basılabilmesi için ödev isteri içerisinde bulunan görünüm fonksiyonunu kullandım ve daha önceden satır sayısını hesapladığımdan gerektiği noktalarda alt satıra geçerek ilk durumu ekrana bastırdım.

Kullanıcıdan giriş almakla birlikte elimde oluşan dizileri birbirini yeme fonksiyonu içerisine yolladım. Farklı bir fonksiyon yazmamın nedeni tamamiyle tek sorumluluk prensibini sağlamaktı. Birbirini yeme içerisinde iki adet iç içe *for* döngüsü kullanarak her bir eleman için farklı bir eleman tarafından yenme durumu oluşana kadar diğer elemanları yeme fikri üzerinden hareket ederek işlemler yaptım. Ne zamanki bir eleman farklı bir eleman tarafından yenirse *for* döngüsü o anki elemanı yiyen elemanın indexine güncellenerek kaldığı yerden devam etti. Buradaki yeme işlemleri için ödev dosyası içerisindeki kuralları takip ettim ve eşitlik ,sayı büyüklüğü durumlarını göz önünde bulundurdum.

Ödevin en zorlayıcı kısımlarından birisi de burası olmakla birlikte, indexlerin takip edilmesi ve doğru atlamaların yapılması algoritmik olarak büyük kazanımlar sağladı. Birbirini yeme işlemini başarılı bir şekilde gerçekleştirdikten sonra her yeme işlemi sonrası, o anki durumu ekrana bastıracak ve son eleman kaldığında sonucu ekrana bastıracak olan durumu göster fonksiyonunu tanımladım. Yine benzer bir *for* döngüsü içeren bu metot ile birlikte her yeme

işlemi sonrası durumu takip ederek doğru sonuca ulaşıp ulaşmadığımı kontrol ettim ve ödev isterindeki gibi sonucu yazdırdım.

Tüm işlemleri başarılı bir şekilde tamamladıktan sonra *heap* içerisinden program dahilinde kullandığım bütün elemanların alanlarını geri iade ederek hiç sızıntı oluşturmadan programı sonlandırdım. Farklı programlarla sızıntı testleri yaptım ve kodumu tekrardan inceleyerek güncelledim. Genel anlamda özellikle kalıtım ve yeme mantığı ödev dahilindeki en zorlayıcı kısımlar olmakla birlikte, sonuç olarak ödev çok büyük kazanımlar sağlayarak beni algoritmik ve nesne yönelimli açıdan çok daha iyi bir noktaya taşıdı.