T.C.
SAKARYA ÜNİVERSİTESİ

BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

PROGRAMLAMA DİLLERİNİN PRENSİPLERİ ÖDEV RAPORU

Uzay Yolculuğu Simülasyonu

B241210300 - Furkan KIRAK

SAKARYA

Nisan, 2025

Programlama Dillerinin Prensipleri Dersi

Uzay Yolculuğu Simülasyonu

Furkan KIRAK

[a] B241210300 2-A

Özet

Uzay Araçlarının bir gezegenden bir gezegene kişileri taşıdığı bir uygulama gerçeklememiz istenmekte.Bunu yaparken kendi zaman sınıfımız ile zamanı yönetip, dosyları txtden okuyup bunları simüle etmemiz bekleniyor. Göz önünde bulundurmamız gereken bazı verilmiş koşullar(Eğer bir uzay aracı yolculuk halinde içinde kimse kalmaz ise IMHA olacaktır, Gezegenlerde tarihler birbirinden farklı olması yanında gezegenlerdeki 1 günün kaç saatten oluştuğu da farklıdır, Simülasyon başlar başlamaz ekranda hedefe varacağı tarih hangi gezegenden gezegene gideceği gibi durumlar gösterilmelidir, Dosyalar verilen verilerden(.txt dosyalarından belirli formatta) okunmalıdır…), bazı bizim uygulamamız gereken durumlar var(Zamanı nasıl yöneteceğimiz(gün ay yıl saat), araç vardıktan sonra imha olmalı mı ? , eğer araç hiç kalkmayacaksa ne olacak?) gibi durumları yönetip kendi yorumlarımızla uygulamayı çözümlememiz beklenmekte. Bu bağlamda kendi yorumlarımla projeyi çözüp istenilenleri tamamladım. Debug işlemleri ve hızlı çıktı için ekstra fonksiyon yazarak çıktıların doğruluğunu karşılaştırdım. Sonuç olarak bu projede öğrendiklerimi gerçek hayatta nasıl projelerde nasıl kullanacağım hakkında bilgi sahibi oldum.

Anahtar Kelimeler: Zaman,IMHA,Gezegen,Araç,DosyaOkuma,Ömür

1. GELİŞTİRİLEN YAZILIM

Zorlandığım kısımlar: çıktıdaki düzeni ayarlamak, varacağı tarihi simülasyon başında hesaplamak, hata kontrolleri, imha olma durumunun yönetilmesi aşağıda bu durumlara nasıl yaklaştığımı ve çözüme kavuşturduğum durumlar detaylı olarak anlatılmaktadır.

Projeye başlarken önce projeyi analiz edip adım adım başladım. Öncelikli sınıflar "Kişi-Gezegen-UzayAraçları" bu sınıflardan dosya okuma için "DosyaOkuma" sınıfı, Zamanı yönetmek ve kurallarını belirlemek için "Zaman" Sınıfı ve bunları Simüle etmek için "Simulasyon" sınıfı. Sırayla gidecek olursak:

Kişi sınıfını oluştururken txtden okuyacağımız parametrelerle sınıfı oluşturacağımız için kurucu metotta tanımladım ve proje nesneye yönelimli olduğu için encapsulation gereği olan parametreleri set ve get fonksiyonlarıyla erişimlerini belirledim. Set fonksiyonlarıyla kontrollerimi sağladım ve son olarak kişinin zaman geçtikçe kalanOmur parametresiyle hayatta olup olmadığını belirleyen metodunu yazdım.

Gezegen ve uzay araçları sınıfı için bunların ortak bir soyut sınıf olan UzayNesnesi sınıfından kalıtım aldırdım aynı özelliklerini bir arada işlemek ve nesne yönelimli yazılımın özelliği olan kalıtımı bu bölümde gerçekledim.

Gezegen sınıfını oluştururken gerekli parametreleri kurucu metottan aldım kalıtım aldığı sınıfın kurucusunda gerçeklememiz gereken nesneyi süper ile gerçekledim ve son olarak saatlik güncelleme fonksiyonu ile gezegendeki zamanı döngü boyunca 1 saat uzatacağımızdan 1 saat uzatmaya ayarlardım.

Uzay araçları sınıfını oluştururken öncelikle kalıtımı gerçekledim daha sonra uzay aracı oluşturmak için gerekli parametreleri kurucuda tanımladım, öncelikle Programı sonsuz döngüye yada simülasyon süresini gereksiz uzunluklara sokacağı durumları yönetmek adına anındaImhaEt metodu tanımladım bu metod ile kontrol sınırlarını anlamsız verilere ulaştığı anlarda gerekli görülürse imha edilecek. Uzay aracının davranışlarını belirlerken uzay aracının içine: yolcu ekleme, hareket etme, araçtaki yolcuların durumlarının saatlik güncellemesi, gibi genel davranışsal metodlarını yazdım. Araç zamanı gezegen zamanına eşit olunca harekete geçecek eğer varana kadar içindeki insanlar ölmezse varış tamamlanır aksi durumda araç imha olur. Eğer varmışsa görevini tamamlamıştır imha olmasına gerek yoktur.

DosyaOkuma sınıfını oluştururken verilen txtdeki verileri # olan kısımlardan parçalara ayırarak her bir parçayı okuduğu txtye göre kurucusuna parametre olarak göndererek nesnelerin oluşturulmasını sağlayan static liste metodlarını yazdım. Bunlar sabit olduğu için public static şeklinde bellekte yerini belirledim çünkü heryerden ulaşılacak sabit bilgilerdir bunlar. Bu işlemleri uzayAraci kişiler ve gezegen için ayrı ayrı listeler oluşturup bu listelerde tutarak oluşturacak metodları ve gerekli hata kontrollerini oluşturdum.

Zaman sınıfını oluştururken karşılaştırmalarda buluncağım için javanın kendiside bulunan compare arayüzünü kalıtım aldım arayüz gerçeklemesi kullandım. Kendi zamanımız kendi kurallarımız üzerinde gideceimiz için öncelikle kendi takvimimi oluşturdum. Tüm ayların gün sayısını belirledim. Belirlediğim tarih standart takvim aslında fakat bunu kendi zaman sınıfımda kendim belirlemiş olarak zamanın nasıl yönetileceğinin kararını verdim. Kurucu metoduna doğrudan bir tarihi string olarak ve gezegendgunsaatsayisini aldım ve bir tane kopyalama constructurı oluşturdum. Her gezegendeki gün sayısı farklı olduğu için bunu gezegenler için yönetirken bu şekil de yönetme kararı aldım, diğer durumlar için de kopyalama constructorı ile zamanı yönettim. Kurucuda tarihi string olarak aldığım için bunu ayrıştırmam gerekiyordu efektif kullanım için bunun için bir fonksiyon yazip bu şekilde yönettim. Programda hata kontrolü olarak ne olur ne olmaz diye eğer hatalı bir tarih girilirse bunu otomatik 2000yılına atamayı sağladım default atama yaptım kısacası. Zamana saat eklenince ne olcağının ve sınır kontrollerini yaptım. Eklenen saat gezegendeki gün sayısına eşit olunca gün atlamasını sağlayan fonksiyonu gün sayısı bulunduğu ayın gün sayısına eşit olunca ay atlamasını ay sayısı 12ye eşit olunca yıl sayısınının arttırılmasını sağlayan tarih ekleme fonksiyonlarını her bir durum için ayrı ayrı yazdım , tarihleri karşilaştirma aralarındaki farkı hesaplama gibi durumları sağlayan fonksiyonları yazdım. Simülasyon başladığında direkt varacağı zamanı hesaplamasını sağlayacak fonksiyonu yazdım. Ve zamanı yönetirken lazım olan compare metotlarını overrride ettim projeme göre gerekli şekilde karşılaştırmayı sağlayan parametleri karşılaştırdım.

Simülasyon sınıfı tüm olayın sürecin işlendiği sınıf ,bu sınıfı tasarlarken 2 farklı başlatma seçeneği ile tasarladım 1.si normal süreç içerisinde çıktı alma 2.si hızlı mod bu modu direkt çıktıyı alarak yaptığım işlemlerde yaptığım varsa hataları hızlıca kontrol etmemi sağlaması için oluşturdum. Programı başlatacak diyebilceğimiz programın ilk aşaması olan kurulumyap ile işlemleri başlatıyoruz. Sınıfları oluşturmak için gerekli verileri txtden okuyup tabiri caizse start butonuna basıyoruz. Bu metotta okuma işlemleri tamamlandıktan sonra arlarındaki bağlantılarda kuruluyor (araçlara yolcuları yerleştir, varacağı tarihi hesapla…). Varacağı tarihi hesaplarken adım adım hesaplıyoruz önce aracın kalkış zamanı ile bulunduğu gezegenin başlama zamanı arasındaki farkı bulunduğu gezegenin 1gününün kaç saat olduğu üzerinden hesaplıyoruz aldığımız bu farkı saat cinsinden tutuyoruz. Daha sonra uzay aracının uçuş süresi ile topluyoruz bu süreyi varış gezegeninin başlangıç saatine ekleyerek varış gezegenin 1gününün kaç saat olduğu verisine göre ekleyip hedefe varacağı tarihi hesaplıyoruz. Sonrasında döngüyü başlatıyoruz döngü her saatte bir güncelleneceği için gezegenlerin ve uzay aracının saatlik güncellemesini yapıyoruz döngü boyunca her saaate bir aynı zamanda ekranı temizliyor ve durumu güncelle fonksiyonu ile ekranı tazeleyip ekranda durumu gösteriyoruz.

Program sınıfını ise basit tutarak sadece başlatıcı olarak hangi modda başlatmak istiyorsak onun önündeki // kaldırarak diğerini aktifleştirerek yaptım. Bu durum programa dahil olmadığı ve sadece debug için kullandığım için bunu program başında sormayı doğru bulmadım. Bu sebeple hızlı modun önünde default olarak yorum satırı kullandım.

2. ÇIKTILAR

Çıktıları kontrol etmek için farklı durumları oluşturan farklı txt setlerini denedim. Durum örneklerinden yola çıkarak projede revizelerde bulundum.

Eğer bir uzay aracı daha varmadan içindeki insanlar ölürse uzay aracı imha olacak.

Peki vardıktan sonra insanlar ölürse ne olacak bu durumda ben uzay aracı görevini tamamladığı için yani gezegenden gezegene uçtuğu için yolcuları gezegene indirmiş ve arasındaki bağ kopmuş olacak dolaysıyla gezegende bu kişiler ölürse uzay aracının imha olmaması gerekir.

Peki gezegendeki başlangıç saati içerisinde bulunduğu uzay aracının saatinden ileriyse uzay aracı hiç kalkmayacak ve kalkışı tamamlamadığı içinde döngü herkes ölene kadar (yada araçtaki insanlar ölene kadar) sonlanmayacak. Bu da bir problem oluşturuyor. Dolaysıyla durum kontrol edilir eğer bu durum oluşuyorsa araç direkt imha edilir. Fakat içerisindeki insanlar öldürülmez çünkü uzay aracı henüz kalkmadığı için insanlar içine daha binmemiştir bu sebeple sadece araç imha edilir.

3. SONUÇ

Sonuç olarak yaptığımız projede java da dosya okuma sınıfını kullandık. Dosyaokuma işlemlerini ve bunların birbiriyle ilişkisi hakında fikir sahibi olduk. Süreç yönetimi yaptığımız için gerçek hayatta bir süreç yönetiminde nasıl davranacağımız konusunda da bilgi sahibi olduk. Uzay Yolculuğu Simülasyonu projesinde bir takip sistemi kullandık buda gerçek hayatta mutlaka karşısımıza çıkacak bi durum farklı olasıklıklara farklı bakış açıları getirdik buda yine gerçek hayatta karşımıza çıkacak durumlara nasıl farklı öneriler getireceğimiz konusunda ufkumuzu açtı. Farklı gezegenlerde farklı zaman olması durumu doğrudan bu şekile gelmese bile gerçek hayatta takibini yapacağımız bir projede bir nesnenin bulunduğu yerden diğer yere geçtiğinde aynı nesnenin 2 farklı yerde farklı sonuçlar doğuracağı yaklaşımını nasıl değerlendirmemiz gerektiği hakkında bakış açısı kazandırdığını düşünüyorum.