T.C.
SAKARYA ÜNİVERSİTESİ

BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

PROGRAMLAMA DİLLERİNİN PRENSİPLERİ ÖDEV RAPORU

Koloni Savaşları

B201210078- Mustafa BİÇER

SAKARYA

Mayıs, 2023

Programlama Dillerinin Prensipleri Dersi

Koloni Savaşları

Mustafa BİÇER

[a] *B201210078 1B Grubu*

Özet

Bize verilen proje ödevi, dışarıdan girdi almak usulü ile çalışacak bir koloni savaştırma oyunudur. Oyun dışarıdan aldığı sayısal veriler ile belli kurallar dahilinde koloniler yaratmakta, bunları tur bazlı bir sistemde savaştırmakta ve genişletmektedir. Ödevin isterleri karşılamasının yanında asıl amacının nesneye dayalı programlama prensiplerine göre tasarlanmamış bir dil ortamında bahsi geçen prensibin gerçeklemesinin (benzetiminin) yapılarak prensibin daha doğru kavranması olduğunu düşünüyorum. Bu ödevde projenin yarattığı en büyük problem benzetimin gerçeklenmesinden ziyade C dilinin varsayılan olarak bazı veri yapılarına sahip olmayışı ve buna bağlı olarak birtakım yapılar arasındaki iletişim ve paylaşım; yapıların kendisinde ise çok biçimliliğe müsait olamama problemi vardı ve beni zorladı. Bu problemlerin çözümü eksik veri yapılarının tek tek kodlanarak projeye işlenmesi ve C dilindeki hakimiyetin artmasıyla bazı dil inceliklerinin öğrenilmiş olması sonucu çözülmüştür. Bu dil incelikleri bellek yönetiminin, işaretçi kullanımının programcıya sağladığı özgürlükten gelmektedir. Son olarak şunu belirtmek istiyorum, 2. Sınıfta (2022) aldığım Veri Yapıları dersinin kazanımlarının faydalarını bu projede gerekli veri yapılarını kodlarken fazlaca gördüm.

Anahtar Kelimeler: Modüler, Benzetim, Veri Yapısı, Nesneye Dayalı Programlama

1. GELİŞTİRİLEN YAZILIM

Geliştirdiğim yazılım “colony_war_game”, C dilinde benzetim usulü kodlanması gerektiği için ve C dilinin fonksiyonel yapısına alışık olmayıp hafıza yönetiminde problemler yaşadığım için proje sürecinde pek çok hata yaptım. Bu hataların üstesinden proje kaynak kodundaki “utils” dosyası altında görülebilecek olan bazı geliştirme zamanı yapılarını kodlayarak geldim. Proje isterlerinden olan modülerliği sağlamak konusunda bir problem yaşadığımı söyleyemem. Bu proje de dahil ödev olarak verilen çoğu projede ve mobil platformlara ürettiğim yazılımlarda temiz bir dosyalama mimarisi ve modüler bir yazılım geliştirmek her zaman ilk prensibim olmuştur. Bu PDP 2. Ödevini yaparken zorlandığımı söyleyemem ancak çok zaman harcadığımı söyleyebilirim. Nasıl daha iyi yaparım? Yazdığım kod nasıl daha okunaklı olur? Projemde bu veri yapısına ihtiyacım bunu kodlasam ve kullansam projemin daha zengin ve derli toplu olacağını biliyorum, bu yeni özellik için zaman ayırmalı mıyım? Gibi birçok soru ve fikir proje sürecinde beynimin içinde şimşek atıyordu. Bu soruların neredeyse hepsine yapıcı bir çözüm buldum, sürecin detaylı takibini Github hesabımdan yapabilirsiniz. Son proje teslim tarihinde proje erişimini herkes olarak değiştireceğim. Anlatımım klasör mimarisi ile başlayıp birtakım yapıların neden yaratıldığı hangi problemi çözdüğü bahsi ile devam edecektir. Projenin omurgası: bin, doc, include, lib, src, makefile, Readme şeklindedir.

Tablo 1. Oyun klasör mimarisi

Colony_war_game	Klasör içeriği
Bin	Main.exe
Doc	B201210078.docx
İnclude	Proje bileşenleri .h dosyaları
Lib	Proje bileşenleri .o dosyaları
Src	Proje bileşenleri .c dosyaları
Makefile	Derleme ve yürütme konsol sıra komutları dosyası
Readme.md	Beni oku dosyası

1.0 Oyun klasör mimarisi c dilinde yazılmış bir uygulamanın sıradan klasör tasarımıdır

- Bin: .exe dosyalarının bulunduğu çıktı klasörü.
- Doc: .pdf formatındaki dokumanın bulunduğu dizin.
- Include: oyunun çalışması için gerekli kaynak kodlarının .h dosyaları dizini.
- Lib: Oyun kaynak kodlarının derlenmiş çıktısı. o dosyaları dizini.
- Src: Oyun kaynak dosyaları bu dizinde bulunmaktadır.
- Makefile: Kaynak dosyalarının sırasıyla derleme komutlarını içeren uzantısız dosya.
- Readme: Beni oku dosyası.

Tablo 2. Include klasör mimarisi

Klasör	Klasör İçeriği
colony	Colony.h
game	Game.h, GameManager.h
log	Log.h
manufacture	AManufacture.h, Manufacture0.h, Manufacture1.h, Manufacture2.h, Manufactures.h
strategy	AStrategy.h, Strategy0.h, Strategy1.h, Strategy2.h, Strategies.h
ui	UI.h
utils	Bool.h, String.h, ArrayList.h, DataType.h, InputManager.h, DebugPrinter.h

1.1 Include klasör mimarisi

a. colony

- Colony.h: Colony yapısı altında bir koloni modeli yazılmıştır. İsterleri karşılayacak şekilde alanlar ve metotlar (benzetim fonksiyon) bu modelin içinde tanımlanmıştır -fonksiyonlar yapıcı fonksiyon içinde atanmıştır-

b. game

- Game.h: Game yapısı altında oyunu oyna, oyunu yok et, oyunu incele gibi metotlar (benzetim fonksiyon) bulunmaktadır. Oyunun akışı “play()” metodu ile sağlanmaktadır. Game yapısı yalnız bir alana sahiptir GameManager yapısındaki bu alanda oyun akışı boyunca oyun verilerinin yönetimi sağlanmaktadır.

- GameManager.h: Konsoldan okunan sayılar bir dizi listesi içerisinde GameManager yapısına gelir ve bu yapı içerisinde işlem görürler. Bu işlemler sonucunda koloni yapıları ve listeleri yaratılır. Bu listeler Game sınıfı tarafından yalnız bir çağrı ile kullanılmak üzere GameManager içerisinde bazı metotlarda sıra işlem görürler. Örneğin: "growOrganizer()", kolonilerin nüfus ve yemek stoğu açısından büyümesini sağlar; "battle()", koloniler listesindeki kolonilerin birbirleri ile savaşmasını sağlar. Bu iki metot her tur Game yapısı altındaki oyna "play()" metodu tarafından çağrılır.

c. log

- Log.h: Bir koloninin tur sonundaki anlık durum kaydını tutmak ve görüntüleyebilmek oluşturduğum bir yapı. Her tur sonunda Game yapısının "play()" metodu içerisinde GameManager içindeki "logger()" metodu çağrılır ve koloniler listesindeki tüm kolonilerin kaydı "Log" tek tek yaratılıp bir dizi listesine aktarılır. Bir turdaki tüm logların bulunduğu bu liste GameManager alanı olan başka bir diziler dizisine aktarılır. Son aktarımın yapıldığı diziler dizisi bütün turların log kayıtlarını saklamaktadır.

d. manufacture

- AManufacture.h: Soyut bir üret metodu(benzetim) barındırmaktadır. Her koloni tur sonunda üretim yapabilmektedir ve üretimlerini sağlamak adına bu soyut sınıftan(benzetim) türeyen çocuk sınıflara sahiptirler.

- ManufactureX.h: dosya ismindeki 'X', 0-1-2 gibi numaralardan oluşmaktadır. Farklı üretim sınıfları mevcuttur ve hepsi AManufacture yapısından türemişlerdir. Ortak metotlar(benzetim), "produce()". Soyutlama benzetimi bu kısımda örneklenmiştir.

e. strategy

- Manufacture ile aynı tasarıma sahip bir klasördür. Her koloninin bir savaş metodu vardır bu metot koloniye atanan strateji sınıfının savaş metodunun döndürdüğü savaş gücünü döndürmektedir.

f. ui

- UI.h: Ui yapısı, GameManager bileşeni içerisindeki "toursLogPack" isimli her turun log kaydının tutulduğu dizi listesini bünyesine alarak "uiShow()" isimli metot içerisinde rapor isterlerine uygun formattaki çıktıyı yaratır ve konsola bastırır.

g. utils

- Projeye başlamadan önce kullanmam gereken veri tiplerini belirleyip bu klasör altında konumlandırdım. Proje sürecinde test aşamasında kullandığım "DebugPrinter.h" bazı yapılar da bu dizin altındadır. Main dosyasındaki karmaşıklığı azaltmak adına girdi yönetimini "InputManager.h" dosyasında yaptım. Konsol girdilerini sayısal popülasyon dizi listesine çevirmektedir.

Not: Rapora sığdıramadığım detaylar için Readme.md dosyasına mutlaka bakın. -Mustafa BİÇER