T.C

KOCAELİ SAĞLIK VE TEKNOLOJİ ÜNİVERSİTESİ LİSANS

YAZILIM MÜHENDİSLİĞİ PROGRAMI

PYTHON İLE LİMAN SİMÜLASYONU

HAZIRLAYANLAR:

YUSUF USTAOĞLU 220502003

AHMET EREN ŞENGÜL 220502036

DERS SORUMLUSU:

PROF. DR. HÜSEYİN TARIK DURU

12.12.2023

Yeniköy Mahallesi Ilıca Caddesi No:29, Başiskele,Kocaeli

info@kocaelisaglik.edu.tr kocaelisaglik.edu.tr

1. ÖZET

Bizden istenilen bu ödevde Python dilini kullanarak öncelikle olaylar.csv dosyasındaki tırları uygun olarak birinci istif alanına gönderip ardından oradaki t anında 20 işlem yapan vinç ile bu tırlardaki 20 ila 30 ton ağırlıklı yükleri gemiler.csv dosyasından aldığımız bilgilere dayanarak uygun olanlara yerleştirmemiz isteniyor , uygun gemiler uygun yükler ile %95 ve üzeri doldukları anda limandan ayrılıyorlar ve bu işlem istenilen simülasyon tamamen bitene kadar devam ediyor.

2. GİRİŞ

Bu ödevde bir liman simülasyonu gerçekleştirmeyi amaçlıyoruz. Bunu yapmak için temel sınıflar oluşturduk , temel sınıflar arasında Truck (Tır), Ship (Gemi), Stack (Yığın) ve Harbor (Liman) bulunmaktadır. Bu simülasyon liman operasyonlarını daha etkili şekilde yönetip uygulamak için tasarlandı. Proje verilen csv dosyalarını okuyarak istenilen transferleri gerçekleştirdik.

3. YÖNTEM

```python
class Truck:
    def __init__(self):
        self.file = "olaylar.csv"
        self.truckDict = {}  # Tır Sözlüğü
        self.truckPlateList = []  # Tır Plaka Listesi
        self.truckLoadList = []  # Tır Yük Listesi
        self.truckTimeList = []  # Tır Zaman Listesi
        self.truckCountryList = []  # Tır Ülke Listesi
        self.createTruckDict()
        self.createTruckPlateList()
        self.createTruckLoadList()  # Liste Oluşturmalar.
        self.createTruckTimeList()
        self.createTruckCountryList()
```

Truck sınıfı oluşturduk ve olaylar.csv dosyasından bilgileri alıyoruz createTruckDict, createTruckPlateList, createTruckLoadList, createTruckTimeList, ve createTruckCountryList metotları bu sınıfın içinde tanımlıdır ve bu metotlar, tır verilerini okuyarak ilgili listeleri ve sözlüğü oluşturma yaptık.

```python
    def createTruckDict(self):  # Csv dosyasını sözlüğe çevirme.
        with open(self.file, "r", encoding="utf-8") as file:  # utf-8 özelliği eklendi
            truck = csv.DictReader(file)
            self.truckDict = sorted(truck, key=lambda x: x['geliş_zamanı'])

    def createTruckLoadList(self):  # Tır yüklerini listeye atma.
        for i in self.truckDict:
            self.truckLoadList.append(int(i["yük_miktarı"]))

    def createTruckCountryList(self):  # Tır ülkelerini listeye atma.
        for i in self.truckDict:
            self.truckCountryList.append(i["ülke"])

    def createTruckPlateList(self):  # Tır plakalarını listeye atma.
        for i in self.truckDict:
            plate = i["tır_plakası"]
            plateNumber = plate[-3:]
            self.truckPlateList.append(plateNumber)

    def createTruckTimeList(self):  # Tır zamanlarını listeye atma.
        for i in self.truckDict:
            self.truckTimeList.append(int(i["geliş_zamanı"]))
        self.truckTimeList.sort()
```

Bu metodlar, "Truck" (Tır) sınıfında tır verilerini işlemek için kullanıyoruz örneğin: createTruckDict CSV dosyasındaki tır verilerini sıralı bir sözlüğe çevirir., createTruckLoadList tır yüklerini listeler. createTruckCountryList tır ülkelerini listeler, createTruckPlateList tır plakalarını listeler. createTruckTimeList tır geliş zamanlarını listeler ve sıralar

```python
class Ship:
    def __init__(self):
        self.file = "gemiler.csv"
        self.sortedDict = {}  # Gemi Sözlüğü
        self.shipNameList = []  # Gemi Adları Listesi
        self.shipCapacityList = []  # Gemi Kapasite Listesi
        self.shipLoadList = []  # Geminin %95'lik Kapasite Listesi
        self.createShipNameList()  # Listeleri oluşturma.
        self.createShipDict()
        self.createShipCapacityList()
        self.shipCondition()
```

Ship sınıfı oluşturduk ve gemiler.csv dosyasından bilgileri alıyoruz createShipNameList, createShipDict,createShipCapacityList, shipCondition metotları bu sınıfın içinde tanımlıdır ve bu metotlar, tır verilerini okuyarak ilgili listeleri ve sözlüğü oluşturma yaptık.

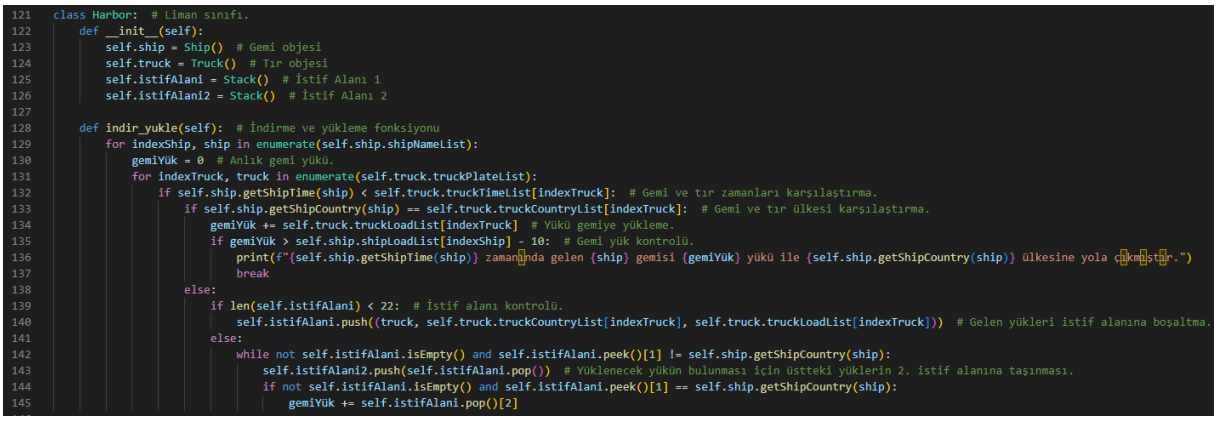
```python
    def createShipDict(self):  # Csv dosyasını sözlüğe çevirme.
        with open(self.file, "r", encoding="utf-8") as file:  # utf-8 özelliği eklendi
            shipDict = csv.DictReader(file)
            self.sortedDict = sorted(shipDict, key=lambda x: x['gemi_adı'])

    def createShipCapacityList(self):  # Gemi kapasitelerini listeye atma.
        for i in self.sortedDict:
            self.shipCapacityList.append(int(i["kapasite"]))

    def createShipNameList(self):  # Gemi isimlerini listeye atma.
        for i in range(1, 451):
            self.shipNameList.append(str(i))

        for index, value in enumerate(self.shipNameList):
            if len(value) == 1:
                self.shipNameList[index] = f"00{value}"
            elif len(value) == 2:
                self.shipNameList[index] = f"0{value}"

    def getShipLoad(self, name):  # Gemi adı ile geminin %95lik kapasitesini alma.
        for i in self.sortedDict:
            if name == i["gemi_adı"]:
                return int(int(i["kapasite"]) * 95 / 100)
```

```python
    def getShipCapacity(self, name):  # Gemi adı ile kapasitesini alma.
        for i in self.sortedDict:
            if name == i["gemi_adı"]:
                return int(i["kapasite"])

    def getShipTime(self, name):  # Gemi adı ile zamanını alma.
        for i in self.sortedDict:
            if name == i["gemi_adı"]:
                return int(i["geliş_zamanı"])

    def getShipCountry(self, name):  # Gemi adı ile ülkesini alma.
        for i in self.sortedDict:
            if name == i["gemi_adı"]:
                return i["gidecek_ülke"]

    def shipCondition(self):  # Gemi %95lik kapasites listesi.
        for i in self.shipCapacityList:
            self.shipLoadList.append(int(i) * 95 / 100)
        return self.shipLoadList
```

Bu metodlar, "Ship" (Gemi) sınıfında gemi verilerini işlemek için kullanıyoruz örneğin: createShipDict CSV dosyasındaki gemi verilerini sıralı bir sözlüğe çevirir. createShipCapacityList gemi kapasitelerini listeler. createShipNameList gemi isimlerini listeler. getShipLoad belirli bir geminin %95'lik kapasitesini hesaplar. getShipCapacity belirli bir geminin tam kapasitesini döndürür. getShipTime belirli bir geminin geliş zamanını döndürür. getShipCountry belirli bir geminin gideceği ülkeyi döndürür. shipCondition ise gemilerin kapasitelerinin %95'ini içeren bir liste oluşturduk.

```python
class Stack:
    def __init__(self):
        self._theItems = list()  # Yığını liste kullanarak yapıyoruz.

    def isEmpty(self):  # Yığının boş olup olmadığını kontrol etme.
        return len(self) == 0

    def __len__(self):  # Yığının uzunluğu
        return len(self._theItems)

    def peek(self):  # En üstteki değere bakma.
        assert not self.isEmpty(), "Boş bir yığında peek işlemi yapılamaz"
        return self._theItems[-1]

    def pop(self):  # Üstteki değeri çıkartma.
        assert not self.isEmpty(), "Boş yığında pop işlemi yapılamaz."
        return self._theItems.pop()

    def push(self, item):  # Değeri yığına ekleme.
        self._theItems.append(item)

    def __str__(self):  # Çıktı vermesi için str metodu.
        return str(self._theItems)
```

Yığını oluşturmak için liste kullanıyoruz , isEmpty ile yığının durumunu kontrol ediyoruz , __len__ ile yığının uzunluğunu aldık , peek ile en üstteki değeri alıp ardından pop ile en üstteki değeri çıkartıyoruz, push ile değeri yığına ekleyip ardından çıktı vermesi için __str__ kullandık.



```python
class Harbor:  # Liman sınıfı.
    def __init__(self):
        self.ship = Ship()  # Gemi objesi
        self.truck = Truck()  # Tır objesi
        self.istifAlani = Stack()  # İstif Alanı 1
        self.istifAlani2 = Stack()  # İstif Alanı 2

    def indir_yukle(self):  # İndirme ve yükleme fonksiyonu
        for indexShip, ship in enumerate(self.ship.shipNameList):
            gemiYük = 0  # Anlık gemi yükü.
            for indexTruck, truck in enumerate(self.truck.truckPlateList):
                if self.ship.getShipTime(ship) < self.truck.truckTimeList[indexTruck]:  # Gemi ve tır zamanları karşılaştırma.
                    if self.ship.getShipCountry(ship) == self.truck.truckCountryList[indexTruck]:  # Gemi ve tır ülkesi karşılaştırma.
                        gemiYük += self.truck.truckLoadList[indexTruck]  # Yükü gemiye yükleme.
                        if gemiYük > self.ship.shipLoadList[indexShip] - 10:  # Gemi yük kontrolü.
                            print(f"{self.ship.getShipTime(ship)} zamanında gelen {ship} gemisi {gemiYük} yükü ile {self.ship.getShipCountry(ship)} ülkesine yola çıkmıştır.")
                            break
                    else:
                        if len(self.istifAlani) < 22:  # İstif alanı kontrolü.
                            self.istifAlani.push((truck, self.truck.truckCountryList[indexTruck], self.truck.truckLoadList[indexTruck]))  # Gelen yükleri istif alanına boşaltma.
                        else:
                            while not self.istifAlani.isEmpty() and self.istifAlani.peek()[1] != self.ship.getShipCountry(ship):
                                self.istifAlani2.push(self.istifAlani.pop())  # Yüklenecek yükün bulunması için üstteki yüklerin 2. istif alanına taşınması.
                                if not self.istifAlani.isEmpty() and self.istifAlani.peek()[1] == self.ship.getShipCountry(ship):
                                    gemiYük += self.istifAlani.pop()[2]
```

Harbor sınıfı oluşturduk ve liman simülasyonunu gerçekleştirmek için gemi ve tır verilerini kullanır. Sınıfın indir_yukle metodu ile gemi ve tır verilerini işleyerek indirme ve yükleme işlemlerini simüle ettik. Gemi ve tır verileri arasında zaman, ülke ve yük miktarı kontrolleri yaparak, gemi yüklerini uygun şekilde işler ve liman operasyonlarını simüle eder. İstif alanlarını kullanarak yük transferini yönetir.

4. SONUÇ VE ÖĞRENİLEN DERSLER

Sonuç olarak elimizde istenilen liman operasyonunu başarılı şekilde simüle eden ve bu iş için kullanışlı bir araç elde ettik. Bu ödevi yaparken karşılaştığımız teknik zorluklar ve bilgilendirme ardında öğrenediğimiz Stack kullanımı gereksinimi ödevi nasıl yapacağımızı baştan değerlendirip Python kodunu tekrar düzenlememize sebep oldu ve bu süreç içerisinde sürekli grup olarak birbirimizle iletişimde olup düzenlemeleri yaptık ayrıca ödevde karşılaşılan kodlama hataları bize tecrübe oldu.

5. KAYNAKÇA

https://www.w3schools.com/python

https://www.youtube.com

6. GİTHUB BAĞLANTILARI

Ahmet Eren Şengül: https://github.com/Eren1213

Yusuf Ustaoğlu: https://github.com/Katlicia