T.C

KOCAELİ SAĞLIK VE TEKNOLOJİ ÜNİVERSİTESİ
MÜHENDİSLİK VE DOĞA BİLİMLERİ FAKÜLTESİ
YAZILIM MÜHENDİSLİĞİ PROGRAMI

ÖDEV KONUSU

LİMAN OTOMASYONU

Hazırlayan

Senem ADALAN - 220502045

Sema Su YILMAZ - 220502016

GitHub Bağlantıları

Senem ADALAN -

Sema Su YILMAZ -

DERS SORUMLUSU

PROF. DR. HÜSEYİN TARIK DURU

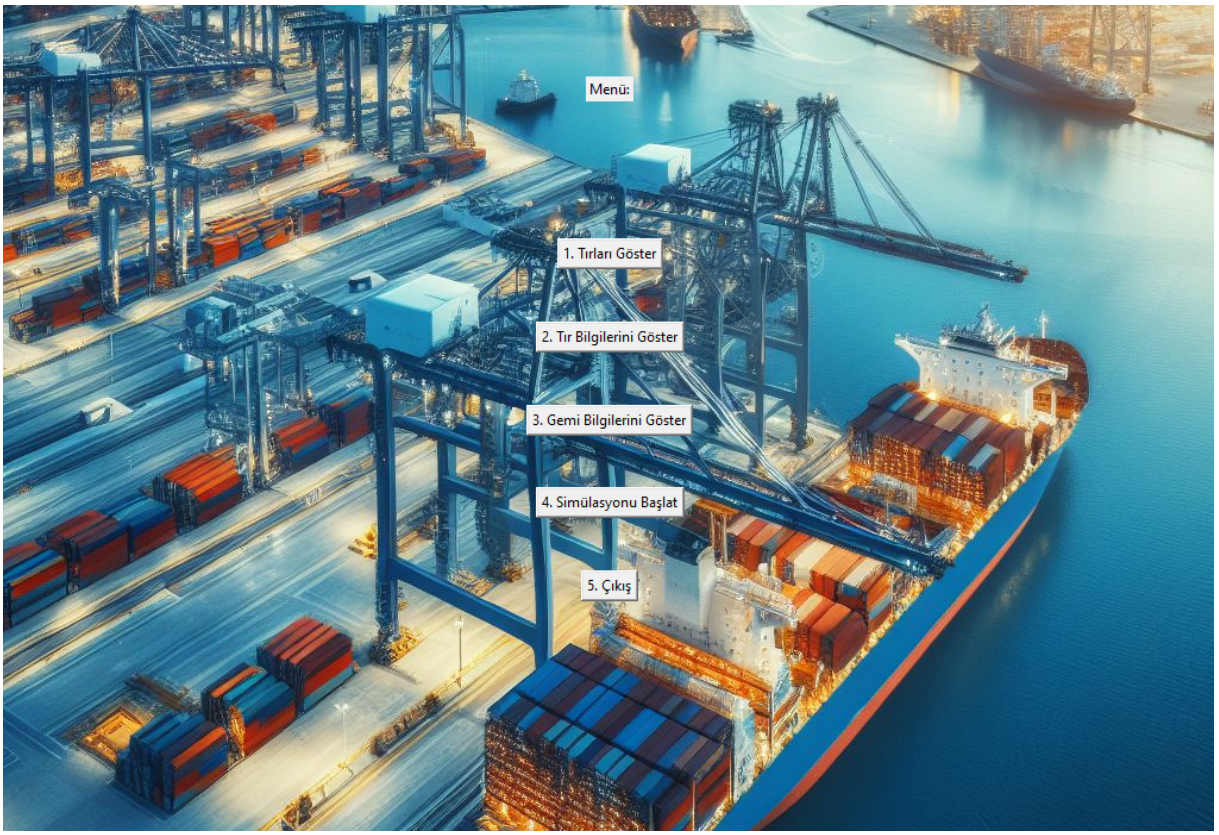
13 ARALIK 2023

İÇİNDEKİLER

1. ÖZET

Bu ödevde Python programlama dilini kullanarak bir limandaki yük indirme-yükleme otomasyon sisteminin simülasyonunu gerçekleştirdik.Bu otomasyonun kullanımı için bir menü oluşturduk ve programımız için arayüz tasarımı yaptık.

2. GİRİŞ



Projenin genel amacı Python programlama dilini kullanarak çeşitli liman işlemlerini simüle eden bir program oluşturmaktır.Kullanıcıların simülasyonu daha kolay bir biçimde kullanabilmeleri için programımıza konsol menü ekledik. Bu menü kullanıcıya farklı işlemleri seçme ve bu işlemleri gerçekleştiren fonksiyonlar kullanma imkanı sunar.

Bu projede işlemler gerçekleştirilirken olaylar.csv ve gemiler.csv dosyaları kullanılmıştır.Bu dosyalardan veriler çekilerek liman otomasyonu için simülasyon gerçekleştirilmiştir.

3. YÖNTEM

Pandas ve Tkinter kütüphaneler içe aktarılır.PhotoImage sınıfı, arayüzdeki arka plana resim eklenmesi için kullanılacaktır. “olaylar.csv” ve “gemiler.csv” dosyalarından veriler Pandas modülü kullanılarak okunur.

```python
import pandas as pd
import tkinter as tk
from tkinter import PhotoImage

# Olaylar.csv ve Gemiler.csv dosyalarını oku
olaylar_df = pd.read_csv("olaylar.csv", encoding='ISO-8859-9')
gemiler_df = pd.read_csv("gemiler.csv", encoding='ISO-8859-9')
```

Tir isminde Python class’ı oluşturulur.Bu Python sınıfı bir “Tır” nesnesini temsil eder.Tir sınıfının özellikleri şunlardır, **plaka**: Tırın plaka numarasını temsil eder, **ulke**: Tırın bulunduğu ülkeyi temsil eder, **ton20**: 20 tonluk yük kapasitesine sahip olan tırın adetini temsil eder, **ton30**: 30 tonluk yük kapasitesine sahip olan tırın adetini temsil eder, **yuk_miktari**: Tırın taşıdığı toplam yük miktarını temsil eder, **maliyet**: Tırın taşıma maliyetini temsil eder, **gelis_zamani**: Tırın limana varış zamanını temsil eder.

```python
class Tir:
    def __init__(self, plaka, ulke, ton20, ton30, yuk_miktari, maliyet, gelis_zamani):
        self.ad = "1000" if plaka == "41_kostu_1000" else plaka[-3:]
        self.plaka = plaka
        self.ulke = ulke
        self.ton20 = ton20
        self.ton30 = ton30
        self.yuk_miktari = yuk_miktari
        self.maliyet = maliyet
        self.gelis_zamani = gelis_zamani
        self.bilgi_sozlugu = {
            'ulke': ulke,
            'ton20_adet': ton20,
            'ton30_adet': ton30,
            'yuk_miktari': yuk_miktari,
            'maliyet': maliyet,
            'yuk': []
        }
```

Gemi isminde Python class'ı oluşturulur.Bu Python sınıfı bir "Gemi" nesnesini temsil eder.Gemi sınıfının özellikleri şunlardır, **adi**: Gemiyi temsil eden ismi, **kapasite**: Geminin taşıma kapasitesi,**gidecek_ulke**: Geminin gideceği ülkeyi temsil eder, **gelis_zamani**: Geminin varış zamanını temsil eder, **yuk**: Geminin taşıdığı yükleri temsil etmek için bir Stack nesnesi oluşturulur, **bilgi_sozlugu**: Gemiye ait çeşitli bilgileri içeren bir sözlük oluşturulur. Bu sözlük, gemiye ait ad, kapasite, gidecek ülke, taşıdığı yüklerin listesi ve toplam yük miktarını içerir.

```
class Gemi:
    def __init__(self, adi, kapasite, gidecek_ulke, gelis_zamani):
        self.adi = adi
        self.kapasite = kapasite
        self.gidecek_ulke = gidecek_ulke
        self.gelis_zamani = gelis_zamani
        self.yuk = Stack()
        self.bilgi_sozlugu = {
            'adi': adi,
            'kapasite': kapasite,
            'gidecek_ulke': gidecek_ulke,
            'yuk': self.yuk.items,
            'yük_miktarı': 0
        }
```

Liman isminde Python class'ı oluşturulur.Bu Python sınıfı bir "Liman" nesnesini temsil eder.Liman sınıfının özellikleri şunlardır, **istif_alani1** ve **istif_alani2**: İstif alanlarını temsil eden iki adet Stack nesnesi oluşturulur. Bu istif alanları tırların ve gemilerin yüklerini depolamak için kullanılır.

max_kapasite: Her bir istif alanının taşıma kapasitesini temsil eder. Bu değer 750 ton olarak belirlenmiştir.

yuk_indir(self, yuk): Bu metod, limana gelen yükleri işler. **yuk** parametresi, indirilecek yükleri temsil eden bir liste veya benzer bir veri yapısıdır. Metodun amacı, yükleri istif alanlarına yerleştirmektir.İlk olarak yuk içindeki tüm yüklerin toplam tonajını hesaplar.Ardından indirilen tonajın limanın belirlediği maksimum kapasiteyi aşıp aşmadığını kontrol eder.Eğer aşmışsa ikinci istif alanına geçilir ve yükler oraya yerleştirilir.

bosalt(self): Bu metod, birinci istif alanındaki yükleri boşaltır. Eğer birinci istif alanında yük varsa en üstteki yükü çıkarır ve kenara alınan yükleri üstüne yerleştirir.

gemileri_beklet(self, t_grubu, gemiler): Bu metod, gelen tırları gemilerle eşleştirmek için kullanılır. Tırın varış zamanı, ülke bilgisi ve geminin varış zamanı gibi kriterlere göre uygun bir gemi bulunur ve tır yüklenir. Eğer uygun bir gemi bulunamazsa tır istif alanında bekletilir.

```python
class Liman:
    def __init__(self):
        self.istif_alani1 = Stack()
        self.istif_alani2 = Stack()
        self.max_kapasite = 750

    def yuk_indir(self, yuk):
        indirilen_tonaj = sum([y['yuk_miktari'] for y in yuk])
        if indirilen_tonaj > self.max_kapasite:
            print(f"İstif alanı kapasitesi aşıldı. İkinci istif alanına geçiliyor.")
            self.istif_alani2.push(yuk)
            print(f"{indirilen_tonaj} ton yük ikinci istif alanına önceki yüklerin üzerine yerleştirildi.","\n")
        else:
            print(f"{indirilen_tonaj} ton yük istif alanına indiriliyor.")
            for y in yuk:
                self.istif_alani1.push([y])  # Her yükü ayrı bir listeye koy
            print(f"{indirilen_tonaj} ton yük birinci istif alanına önceki yüklerin üzerine yerleştirildi.", "\n")
```

```python
    def bosalt(self):
        if not self.istif_alani1.is_empty():
            print("İstif alanı boşaltılıyor.")
            kenara_alinan_yukler = self.istif_alani1.peek()[-1 * t.yuk_miktari:]   # İlgili yükün üstündeki yükleri kenara al
            self.istif_alani1.pop()  # İlk istif alanındaki yükü çıkar
            self.istif_alani1.items += kenara_alinan_yukler  # Kenara alınan yükleri üste yerleştir

            total_load = sum([y['yuk_miktari'] for y in self.istif_alani1.peek()[0]])   # İstif alanındaki toplam yükü hesapla
            print(f"İstif alanındaki toplam yük: {total_load} ton")
        else:
            print("İstif alanı boş.")

    def gemileri_beklet(self, t_grubu, gemiler):
        loading_message_printed = False
        for t in t_grubu:
            uygun_gemi_bulundu = False
            for gemi in gemiler:
                if gemi.gidecek_ulke == t.ulke and gemi.gelis_zamani <= t.gelis_zamani:
                    uygun_gemi_bulundu = True
                    gemi.yuk.push(t.bilgi_sozlugu['yuk'])
                    if not loading_message_printed:
                        print(f"Gemi Yükleniyor:")
                        loading_message_printed = True
                    print(f"   - Gemi: {gemi.adi}, Ülke: {gemi.gidecek_ulke}, "
                          f"Kapasite: {gemi.kapasite} ton, {t.plaka} plakalı Tır yüklendi.")
                    break
```

Stack ismindeki Python sınıfı bir Stack (yığın) veri yapısını temsil eder. Stack veri yapısı, verilerin üst üste konulduğu ve sadece en üstteki verinin erişilebildiği bir yapıdır. Bu sınıfın metodları şu işlevlere sahiptir:

__init__(self): Bu başlatıcı metod Stack sınıfının örneğini oluşturur ve başlangıçta boş bir liste (self.items) ile başlatır.

is_empty(self): Bu metod yığının boş olup olmadığını kontrol eder. Yığın boş ise True, değilse False döndürür.

push(self, item): Bu metod yığının en üstüne yeni bir öğe (item) ekler.

pop(self): Bu metod yığının en üstündeki öğeyi çıkarır ve bu öğeyi döndürür. Eğer yığın boşsa, None değeri döner.

peek(self): Bu metod yığının en üstündeki öğeye erişir ancak çıkarmaz. Yani yığından hiçbir eleman silinmez.

size(self): Bu metod yığındaki öğe sayısını döndürür.

```python
class Stack:    #Yüklerin üst üste istiflenebilmesi için stack sınıfı oluştur
    def __init__(self):
        self.items = []

    def is_empty(self):
        return len(self.items) == 0

    def push(self, item):
        self.items.append(item)

    def pop(self):
        if not self.is_empty():
            return self.items.pop()
        else:
            return None

    def peek(self):
        return self.items

    def size(self):
        return len(self.items)
```

"olaylar.csv" dosyasından okunan verilerle tırlar **tirlar** listesine eklenir ve her bir tır nesnesine ait "gemi" bilgisi başlangıçta None olarak ayarlanır.**tirlar** ve **gemiler** adlı boş listeler oluşturulur.Bir liman nesnesi olan **liman** oluşturulur."olaylar.csv" dosyasındaki her bir satır üzerinde dönen bir döngü başlatılır (**for index, row in olaylar_df.iterrows():**).Her bir satırdan tır bilgileri okunarak yeni bir **Tir** nesnesi oluşturulur ve bu nesne **tirlar** listesine eklenir.Her bir Tır nesnesinin **bilgi_sozlugu** sözlüğüne "gemi" adında bir anahtar eklenir ve başlangıçta bu anahtarın değeri None olarak atanır.

```
#Liman nesnesi ve Tır/Gemi listeleri oluştur
tirlar = []
gemiler = []
liman = Liman()

# Olaylar.csv'den tırların listesini oluştur
for index, row in olaylar_df.iterrows():
    plaka = row['tır_plakası']
    ulke = row['Ülke']
    ton20 = row['20_ton_adet']
    ton30 = row['30_ton_adet']
    yuk_miktari = row['yük_miktarı']
    maliyet = row['maliyet']
    gelis_zamani = row['geliş_zamanı']
    t = Tir(plaka, ulke, ton20, ton30, yuk_miktari, maliyet, gelis_zamani)
    tirlar.append(t)
    t.bilgi_sozlugu['gemi'] = None
```

"gemiler.csv" dosyasındaki her bir satır üzerinde dönen bir döngü başlatılır (**for index, row in gemiler_df.iterrows():**).Her bir satırdan gemi bilgileri okunarak yeni bir **Gemi** nesnesi oluşturulur bu nesne gemiler listesine eklenir.**gemiler** listesi, gemilerin kapasitelerine göre büyükten küçüğe sıralanır (**gemiler.sort(key=lambda x: x.kapasite, reverse=True)**).**tirlar** listesi, tırların geliş zamanlarına ve plaka numaralarına göre sıralanır (**tirlar.sort(key=lambda x: (x.gelis_zamani, x.plaka))**).**gruplar** adlı boş bir sözlük oluşturulur.Tırların listesinde dönülerek her bir tır, geliş zamanına göre gruplanır. Eğer aynı geliş zamanına sahip başka tırlar da varsa aynı anahtar altında toplanır (**gruplar[t.gelis_zamani]**).

```python
for index, row in gemiler_df.iterrows():
    adi = row['gemi_adı']
    kapasite = row['kapasite']
    gidecek_ulke = row['gidecek_ülke']
    gelis_zamani = row['geliş_zamanı']
    g = Gemi(adi, kapasite, gidecek_ulke, gelis_zamani)
    gemiler.append(g)

gemiler.sort(key=lambda x: x.kapasite, reverse=True)   #Gemileri kapasitelerine göre sırala
tirlar.sort(key=lambda x: (x.gelis_zamani, x.plaka))   #Tırları geliş zamanına ve plaka numarasına göre sırala

gruplar = {}
for t in tirlar:
    if t.gelis_zamani in gruplar:
        gruplar[t.gelis_zamani].append(t)
    else:
        gruplar[t.gelis_zamani] = [t]
```

Bu fonksiyon, tirlar listesindeki tır nesnelerini küçükten büyüğe sıralar ve ardından her bir tırın plaka numarasını ve adını ekrana yazdırır.**tirlar_sirali** adlı değişken, tirlar listesindeki tırları tırın adının "1000" olup olmamasına göre sıralayarak elde eder.

```python
def sirali_tirlar():
    tirlar_sirali = sorted(tirlar, key=lambda x: (x.ad == "1000", x.ad))
    print("\nSıralı Tırlar:")
    plakalar = set()
    for tir in tirlar_sirali:
        if tir.plaka not in plakalar:
            print(f"Tır Plakası: {tir.plaka}, Tır Adı: {tir.ad}")
            plakalar.add(tir.plaka)
```

Bu fonksiyon, kullanıcıdan bir tırın plaka numarasını alarak, o tıra ait bilgileri "olaylar.csv" dosyasından çeker ve ekrana yazdırır.Kullanıcıdan bir tırın plaka numarasını içeren bir giriş alınır.(**(next((tir for tir in tirlar if tir.plaka == plaka), None**) ifadesi kullanılarak, tirlar listesindeki tırlar arasında kullanıcının girdiği plakaya sahip tırı bulur. Eğer tır bulunamazsa, None döner.Giriş zamanı, ülke, tonaj bilgileri, yük miktarı ve maliyet bilgileri ayrı ayrı satırlar halinde ekrana yazdırılır.Eğer belirli bir tır bulunamazsa, "Belirtilen Tır bulunamadı." mesajı ekrana yazdırılır.

```python
def tir_bilgileri():
    plaka = input("Hangi Tır'ın bilgilerini görmek istiyorsunuz? (Örneğin: 41_kostu_001): ")
    secili_tir = next((tir for tir in tirlar if tir.plaka == plaka), None)
    if secili_tir:
        print(f"\nTır Plakası: {secili_tir.plaka}, Tır Adı: {secili_tir.ad}")
        girisler = sorted(zip(olaylar_df[(olaylar_df['tır_plakası'] == plaka)]['geliş_zamanı'],
                              olaylar_df[(olaylar_df['tır_plakası'] == plaka)]['ülke'],
                              olaylar_df[(olaylar_df['tır_plakası'] == plaka)]['20_ton_adet'],
                              olaylar_df[(olaylar_df['tır_plakası'] == plaka)]['30_ton_adet'],
                              olaylar_df[(olaylar_df['tır_plakası'] == plaka)]['yük_miktarı'],
                              olaylar_df[(olaylar_df['tır_plakası'] == plaka)]['maliyet']
                              ))

        # Her bir giriş için ayrı satırlar oluştur
        for guncel_gelis_zamani, ulke, ton20_adet, ton30_adet, yuk_miktari, maliyet in girisler:
            print(f"Geliş Zamanı: {guncel_gelis_zamani}")
            print(f"Ülkeler: {ulke}")
            print(f"20_ton_konteyner_adedi: {ton20_adet}")
            print(f"30_ton_konteyner_adedi: {ton30_adet}")
            print(f"Yük_miktarı: {yuk_miktari}")
            print(f"Maliyet: {maliyet}", "\n")
    else:
        print("Belirtilen Tır bulunamadı.")
```

Bu kodda gemi bilgileri fonksiyonu tanımlanır.**gemi_bilgileri** adlı bir fonksiyon tanımlanır.Fonksiyonun içinde, gemi bilgilerini ekrana yazdırmak için bir döngü başlatılır. Döngü, gemiler_df veri çerçevesinin her satırını gezerek çalışır.Her bir satır için, gemi adı **(adi),** kapasite **(kapasite)**, gidecek ülke **(gidecek_ulke)** ve geliş zamanı **(gelis_zamani)** gibi sütun değerleri alınır.

```python
def gemi_bilgileri():
    print("\nGemi Bilgileri:")
    for index, row in gemiler_df.iterrows():
        adi = row['gemi_adı']
        kapasite = row['kapasite']
        gidecek_ulke = row['gidecek_ülke']
        gelis_zamani = row['geliş_zamanı']

        print(f"Gemi Adı: {adi}, Kapasite: {kapasite} ton, Gidecek Ülke: {gidecek_ulke}, Geliş Zamanı: {gelis_zamani}")
```

tirlari_indir() fonksiyonu gruplar adlı sözlükteki her bir geliş zamanı ve ilgili tırlar için bir döngü başlatır.Her bir geliş zamanı için gemiler kontrol edilir. Gemi, gideceği ülke ve geliş zamanı ile eşleşen ilk gemi bulunur ve bu bilgiler ekrana yazdırılır.Ardından belirli bir geliş zamanına ait tırların bilgileri ekrana yazdırılır. Her bir tır için plaka, ülke, tonaj bilgileri ve indirilen tonaj miktarı hesaplanarak ekrana yazdırılır.**liman.gemileri_beklet(t_grubu, gemiler)** fonksiyonu çağrılarak, belirli bir geliş zamanına ait tırların gemilere yüklenmesi işlemi gerçekleştirilir.

cikis() fonksiyonu **root.destroy()** ifadesi, bir GUI uygulamasının ana penceresini kapatmak için kullanılır. root objesi, Tkinter penceresini temsil eder ve destroy() metodu bu pencereyi kapatır.

```
def tirlari_indir():
    for gelis_zamani, t_grubu in gruplar.items():
        print("-----------------------------------------------")
        print(f"Zaman: {gelis_zamani}")

        for gemi in gemiler:
            if gemi.gidecek_ulke == t_grubu[0].ulke and gemi.gelis_zamani == gelis_zamani:
                print(f"Gemi Geliyor: Gemi: {gemi.adi} (Ülke: {gemi.gidecek_ulke}) - Kapasite: {gemi.kapasite} ton")
                break

        print("Tırlar İndiriliyor:")
        for t in t_grubu:
            print(f"   - Tır: {t.plaka}, Ülke: {t.ulke}, 20 ton konteyner: {t.ton20} adet, 30 ton konteyner: {t.ton30} adet")
            total_ton20 = t.ton20 * 20
            total_ton30 = t.ton30 * 30
            total_ton = total_ton20 + total_ton30
            print(f"   - İstif Alanına {total_ton} ton konteyner indiriliyor. (Üst üste, Plaka: {t.plaka})")

        liman.gemileri_beklet(t_grubu, gemiler)

def cikis():
    root.destroy()
```

Tkinter tabanlı bir GUI (Graphical User Interface - Grafiksel Kullanıcı Arayüzü) uygulaması oluşturulur.Tkinter ile liman otomasyon sistemi için bir GUI penceresi oluşturulur.Arka plana liman resmi eklenir.Pencerede "Menü:" metni görüntülenir."Tırları Göster","Tır Bilgileri","Gemi Bilgileri ","Simülasyon Başlat", ve "Çıkış" işlevli butonlar eklenir.Tkinter ana döngüsü başlatılır ve kullanıcı etkileşimine olanak tanır.

```
# Arka plan resmini yükle
resim_image = PhotoImage(file="liman.png")
resim_label = tk.Label(root, image=resim_image)
resim_label.place(relwidth=1, relheight=1)

menu_label = tk.Label(root, text="Menü:")
menu_label.place(relx=0.5, rely=0.1, anchor=tk.CENTER)

# Butonları ekle
sirali_tirlar = tk.Button(root, text="1. Tırları Göster", command=sirali_tirlar)
sirali_tirlar.place(relx=0.5, rely=0.3, anchor=tk.CENTER)

tir_bilgileri = tk.Button(root, text="2. Tır Bilgilerini Göster", command=tir_bilgileri)
tir_bilgileri.place(relx=0.5, rely=0.4, anchor=tk.CENTER)

gemi_bilgileri = tk.Button(root, text="3. Gemi Bilgilerini Göster", command=gemi_bilgileri)
gemi_bilgileri.place(relx=0.5, rely=0.5, anchor=tk.CENTER)

tirlari_indir = tk.Button(root, text="4. Simülasyonu Başlat", command=tirlari_indir)
tirlari_indir.place(relx=0.5, rely=0.6, anchor=tk.CENTER)

cikis = tk.Button(root, text="5. Çıkış", command=cikis)
cikis.place(relx=0.5, rely=0.7, anchor=tk.CENTER)

root.mainloop()
```

4. SONUÇ VE ÖĞRENİLEN DERSLER

Bir projeyi geliştirirken yazılım aşamalarını takip etmemiz gerektiğini öğrendik.Özellikle artırımlı geliştirmenin bu aşamada oldukça önemli olduğunu fark ettik.Bir simülasyon yazarken birçok işlem bir arada gerçekleştirildiği için program yazarken çok daha dikkatli olmamız gerektiğini kavradık.Önceden kullanmadığımız Tkinter kütüphanesini kullanarak programımıza arayüz eklemeyi öğrendik.Bu arayüzü programımıza eklerken bir yapay zekaya çizim yaptırmayı ve bu çizimi programımızdaki arayüze eklemeyi öğrendik.Uzun soluklu projelerde pes etmeden sabırla ilerlemeyi öğrendik.Bu tarz kapsamlı projelerde düzenli,disiplinli ve fazla çalışmamız gerektiğini anladık.

5. KAYNAKÇA

https://www.youtube.com/watch?v=ioNfInaP-EA&list=PLSmHiN0iazy_qX_6Tmecj4tTOefqh2-m2

https://www.youtube.com/watch?v=MUihiqjvy2I&list=PLwBOFNYr7HY9E8sfsEXu4wuJW_PbRZQS6

https://www.youtube.com/watch?v=Ax4iI-2Ktp0&list=PL-Hkw4CrSVq-qwoZJKGXdYoGYQseJoPR2

https://www.youtube.com/watch?v=IY0W0FmqBfo&list=PLfMRLSpipmfsLoyO-deGWkJ0RAQf9gU20

https://www.codingtxt.com/blog/detail/python-tkinter-arayuz-programlama-temel-bilgiler-ve-ornekler-61208d72-5a59-43#gsc.tab=0

https://www.veribilimiokulu.com/python-ile-makine-ogrenmesine-giris-pandas-kutuphanesi/

https://chat.openai.com

Veri Yapıları Ders Notları — Nur Banu ALBAYRAK