Araba kiralama

18.32 kiralama bitiri yapmamaya karar verdim

import tkinter as tk  
from tkinter import ttk, simpledialog, messagebox  
import sqlite3  
import datetime  
from abc import ABC, abstractmethod  
class Araba(ABC):  
 def \_\_init\_\_(self, marka, model, yil):  
 self.id = None  
 self.marka = marka  
 self.model = model  
 self.\_yil = yil # Protected özellik  
 self.\_\_kilometre = 0 # Private özellik  
  
 def get\_yil(self):  
 return self.\_yil  
  
 def set\_kilometre(self, km):  
 if km >= 0:  
 self.\_\_kilometre = km  
 else:  
 print("Geçersiz kilometre değeri.")  
  
 def get\_kilometre(self):  
 return self.\_\_kilometre  
  
 @abstractmethod  
 def bakim(self):  
 pass  
  
class ElektrikliAraba(Araba):  
 def \_\_init\_\_(self, marka, model, yil, batarya\_durumu):  
 super().\_\_init\_\_(marka, model, yil)  
 self.batarya\_durumu = batarya\_durumu  
  
 def bakim(self):  
 return f"{self.marka} {self.model} elektrikli araba bakımı: Pil sağlığı kontrol ediliyor. Batarya Durumu: {self.batarya\_durumu}"  
  
class BenzinliAraba(Araba):  
 def \_\_init\_\_(self, marka, model, yil, yakit\_durumu):  
 super().\_\_init\_\_(marka, model, yil)  
 self.yakit\_durumu = yakit\_durumu  
  
 def bakim(self):  
 return f"{self.marka} {self.model} benzinli araba bakımı: Yağ değiştiriliyor. Yakıt Durumu: {self.yakit\_durumu}"  
class Kiralama:  
 def \_\_init\_\_(self, musteri, araba, kiralama\_tipi, sure):  
 self.musteri = musteri  
 self.araba = araba  
 self.kiralama\_tipi = kiralama\_tipi  
 self.sure = sure  
  
 def hesapla\_ucret(self):  
 if self.kiralama\_tipi == "günlük":  
 return self.sure \* 1000  
 elif self.kiralama\_tipi == "saatlik":  
 return self.sure \* 500  
 else:  
 print("Geçersiz kiralama tipi")  
 return -1  
  
  
class AracKiralamaUygulamasi:  
 def \_\_init\_\_(self, root):  
 self.root = root  
 self.root.title("Araç Kiralama Uygulaması")  
  
 self.create\_database\_connection()  
  
 self.create\_main\_frame()  
  
 def create\_database\_connection(self):  
 self.baglanti = sqlite3.connect("rent-a.db")  
 self.cursor = self.baglanti.cursor()  
  
 def create\_main\_frame(self):  
 self.main\_frame = ttk.Frame(self.root, padding="20")  
 self.main\_frame.grid(row=0, column=0, sticky=(tk.W, tk.E, tk.N, tk.S))  
  
 self.label\_welcome = ttk.Label(self.main\_frame, text="Araç Kiralama Uygulamasına Hoş Geldiniz!", font=('Arial', 16, 'bold', "italic"))  
 self.label\_welcome.grid(row=0, column=0, columnspan=3, pady=20)  
  
 self.label\_instruction = ttk.Label(self.main\_frame, text="Adınız Soyadınız:", font=('Arial', 12))  
 self.label\_instruction.grid(row=1, column=0, columnspan=3, pady=10)  
  
 self.entry\_ad\_soyad = ttk.Entry(self.main\_frame, font=('Arial', 12))  
 self.entry\_ad\_soyad.grid(row=2, column=0, columnspan=3, pady=10)  
  
 # Giriş yap butonu ve stili  
 self.button\_giris = tk.Button(self.main\_frame, text="Giriş Yap", command=self.musteri\_giris,  
 width=10, height=1, bg="purple", fg="white", font=("Arial", 12))  
 self.button\_giris.grid(row=3, column=0, columnspan=3, pady=20)  
  
 def musteri\_giris(self):  
 ad\_soyad = self.entry\_ad\_soyad.get()  
  
 if not ad\_soyad:  
 messagebox.showwarning("Uyarı", "Lütfen adınızı ve soyadınızı girin.")  
 return  
  
 musteri\_id = self.add\_musteri(ad\_soyad)  
 if musteri\_id:  
 self.musteri\_id = musteri\_id  
 self.label\_welcome.config(text=f"Hoş Geldin, {ad\_soyad}!")  
 self.label\_instruction.grid\_remove()  
 self.entry\_ad\_soyad.grid\_remove()  
 self.button\_giris.grid\_remove()  
  
 self.show\_arac\_list()  
  
 def add\_musteri(self, ad\_soyad):  
 self.cursor.execute("INSERT INTO musteri (ad\_soyad) VALUES (?)", (ad\_soyad,))  
 self.baglanti.commit()  
 return self.cursor.lastrowid  
  
 def show\_arac\_list(self):  
 araclar = self.get\_arac\_list()  
  
 self.tree\_araclar = ttk.Treeview(self.main\_frame, columns=("id", "Marka", "Model", "Yıl", "Tip", "Durum"), show="headings")  
 self.tree\_araclar.heading("id", text="id")  
 self.tree\_araclar.heading("Marka", text="Marka")  
 self.tree\_araclar.heading("Model", text="Model")  
 self.tree\_araclar.heading("Yıl", text="Yıl")  
 self.tree\_araclar.heading("Tip", text="Tip")  
 self.tree\_araclar.heading("Durum", text="Durum")  
  
 for arac in araclar:  
 self.tree\_araclar.insert("", "end", values=arac)  
  
 self.tree\_araclar.grid(row=1, column=0, columnspan=3, sticky=(tk.W, tk.E))  
  
 self.button\_kirala = ttk.Button(self.main\_frame, text="Araç Kirala", command=self.arac\_kirala)  
 self.button\_kirala.grid(row=2, column=0, columnspan=3, pady=10, sticky=(tk.W, tk.E))  
  
 def get\_arac\_list(self):  
 self.cursor.execute(  
 "SELECT arabalar.id, marka, model, yil, tip, durumu FROM arabalar WHERE durumu IN ('müsait', 'kiralandı')")  
 return self.cursor.fetchall()  
  
 def arac\_kirala(self):  
 selected\_item = self.tree\_araclar.selection()  
 if selected\_item:  
 arac\_id = self.tree\_araclar.item(selected\_item, "values")[0]  
 arac\_durumu = self.tree\_araclar.item(selected\_item, "values")[5] # Assuming status column is the sixth index  
  
 if arac\_durumu == 'kiralandı':  
 messagebox.showwarning("Uyarı", "Bu araç zaten kiralanmış!")  
 return  
  
 kiralama\_tipi = self.show\_kiralama\_tipi\_dialog()  
 if kiralama\_tipi:  
 sure = self.show\_kiralama\_sure\_dialog(kiralama\_tipi)  
  
 if sure:  
 arac = self.get\_arac\_by\_id(arac\_id)  
  
 # Aracın bakım durumu GUI'de gösteriliyor  
 self.show\_arac\_bakim\_durumu(arac)  
  
 # Aracın durumunu güncelle  
 self.update\_arac\_durumu(arac\_id)  
  
 # Kiralama bilgilerini veritabanına ekle  
 kiralama = Kiralama(self.musteri\_id, arac, kiralama\_tipi, sure)  
 self.add\_kiralama\_bilgisi(kiralama)  
  
 ucret = kiralama.hesapla\_ucret()  
 if ucret != -1:  
 messagebox.showinfo("Başarılı", f"Araç kiralandı! Ücret: {ucret} TL")  
 self.root.destroy()  
  
 def update\_arac\_durumu(self, arac\_id):  
 self.cursor.execute("UPDATE arabalar SET durumu = 'kiralandı' WHERE id = ?", (arac\_id,))  
 self.baglanti.commit()  
  
 def add\_kiralama\_bilgisi(self, kiralama):  
 ucret = kiralama.hesapla\_ucret() # Calculate the rental fee  
  
 self.cursor.execute(  
 "INSERT INTO kiralama (musteri\_id, araba\_id, kiralama\_tipi, sure, ucret, durumu) VALUES ( ?, ?, ?, ?, ?, ?)",  
 (kiralama.musteri, kiralama.araba.id, kiralama.kiralama\_tipi, kiralama.sure, ucret, 'kiralandı'))  
 self.baglanti.commit()  
  
 def show\_arac\_bakim\_durumu(self, arac):  
 bakim\_durumu = arac.bakim()  
 messagebox.showinfo("Bakım Durumu", bakim\_durumu)  
  
 def show\_kiralama\_tipi\_dialog(self):  
 while True:  
 kiralama\_tipi = simpledialog.askstring("Kiralama Tipi", "Kiralama Tipini Seçiniz (günlük/saatlik)")  
  
 if kiralama\_tipi is None:  
 return None # Eğer kullanıcı iptal ederse  
  
 kiralama\_tipi\_lower = kiralama\_tipi.lower()  
 if kiralama\_tipi\_lower not in ["günlük", "saatlik"]:  
 messagebox.showwarning("Uyarı", "Lütfen 'günlük' veya 'saatlik' olarak girin.")  
 else:  
 return kiralama\_tipi\_lower  
  
 def show\_kiralama\_sure\_dialog(self, kiralama\_tip):  
 while True:  
 if kiralama\_tip == "günlük":  
 message = "Günlük Kiralama Süresini Giriniz (gün)"  
 elif kiralama\_tip == "saatlik":  
 message = "Saatlik Kiralama Süresini Giriniz (saat)"  
  
 kiralama\_suresi = simpledialog.askinteger("Kiralama Süresi", message)  
  
 if kiralama\_suresi is None:  
 return None # Eğer kullanıcı iptal ederse  
  
 if kiralama\_suresi <= 0:  
 messagebox.showwarning("Uyarı", "Lütfen pozitif bir değer girin.")  
 else:  
 return kiralama\_suresi  
  
 def get\_arac\_by\_id(self, arac\_id):  
 self.cursor.execute("SELECT \* FROM arabalar WHERE id = ?", (arac\_id,))  
 arac\_data = self.cursor.fetchone()  
  
 if arac\_data:  
 arac\_id, marka, model, yil, tip, durumu,kilometre = arac\_data  
 if tip == 'Elektrikli':  
 elektrikli\_arac = ElektrikliAraba(marka, model, yil, batarya\_durumu="Yüksek")  
 elektrikli\_arac.id = arac\_id # Set the id attribute  
 return elektrikli\_arac  
 elif tip == 'Benzinli':  
 benzinli\_arac = BenzinliAraba(marka, model, yil, yakit\_durumu="Doluluk Oranı: %100")  
 benzinli\_arac.id = arac\_id # Set the id attribute  
 return benzinli\_arac  
 else:  
 return None  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 root = tk.Tk()  
 app = AracKiralamaUygulamasi(root)  
 root.mainloop()

import sqlite3  
  
baglanti = sqlite3.connect("rent-a.db")  
cursor = baglanti.cursor()  
  
cursor.execute('''  
 CREATE TABLE IF NOT EXISTS musteri (  
 id INTEGER PRIMARY KEY AUTOINCREMENT,  
 ad\_soyad TEXT  
 )  
''')  
  
# Araba tablosunu oluştur  
cursor.execute('''  
 CREATE TABLE IF NOT EXISTS arabalar (  
 id INTEGER PRIMARY KEY AUTOINCREMENT,  
 marka TEXT,  
 model TEXT,  
 yil INTEGER,  
 tip TEXT,  
 durumu TEXT DEFAULT 'müsait'  
 )  
''')  
  
# Kiralama tablosunu oluştur  
cursor.execute('''  
 CREATE TABLE IF NOT EXISTS kiralama (  
 musteri\_id INTEGER,  
 araba\_id INTEGER,  
 kiralama\_tipi TEXT,  
 sure INTEGER,  
 ucret INTEGER,  
 durumu TEXT DEFAULT 'kiralandı',  
 FOREIGN KEY (araba\_id) REFERENCES arabalar (id),  
 FOREIGN KEY (musteri\_id) REFERENCES musteri(id)  
 )  
''')  
  
  
  
# Elektrikli Arabalar  
cursor.execute("INSERT INTO arabalar (marka, model, yil, tip) VALUES ('Tesla', 'Model S', 2023, 'Elektrikli')")  
cursor.execute("INSERT INTO arabalar (marka, model, yil, tip) VALUES ('Porsche', 'Taycan', 2023, 'Elektrikli')")  
cursor.execute("INSERT INTO arabalar (marka, model, yil, tip) VALUES ('Audi', 'e-tron', 2023, 'Elektrikli')")  
cursor.execute("INSERT INTO arabalar (marka, model, yil, tip) VALUES ('Mercedes-Benz', 'EQC', 2023, 'Elektrikli')")  
cursor.execute("INSERT INTO arabalar (marka, model, yil, tip) VALUES ('Jaguar', 'I-PACE', 2023, 'Elektrikli')")  
  
# Benzinli Arabalar  
cursor.execute("INSERT INTO arabalar (marka, model, yil, tip) VALUES ('Mercedes-Benz', 'S-Class', 2023, 'Benzinli')")  
cursor.execute("INSERT INTO arabalar (marka, model, yil, tip) VALUES ('BMW', '7 Series', 2023, 'Benzinli')")  
cursor.execute("INSERT INTO arabalar (marka, model, yil, tip) VALUES ('Lexus', 'LS', 2023, 'Benzinli')")  
cursor.execute("INSERT INTO arabalar (marka, model, yil, tip) VALUES ('Audi', 'A8', 2023, 'Benzinli')")  
cursor.execute("INSERT INTO arabalar (marka, model, yil, tip) VALUES ('Porsche', 'Panamera', 2023, 'Benzinli')")  
  
baglanti.commit()  
# Veritabanı bağlantısını kapat  
baglanti.close()

18.09

Düzenlendi fakat kiralama bitir hala eklenmedii

import tkinter as tk  
from tkinter import ttk, simpledialog, messagebox  
import sqlite3  
import datetime  
from abc import ABC, abstractmethod  
class Araba(ABC):  
 def \_\_init\_\_(self, marka, model, yil):  
 self.id = None  
 self.marka = marka  
 self.model = model  
 self.\_yil = yil # Protected özellik  
 self.\_\_kilometre = 0 # Private özellik  
  
 def get\_yil(self):  
 return self.\_yil  
  
 def set\_kilometre(self, km):  
 if km >= 0:  
 self.\_\_kilometre = km  
 else:  
 print("Geçersiz kilometre değeri.")  
  
 def get\_kilometre(self):  
 return self.\_\_kilometre  
  
 @abstractmethod  
 def bakim(self):  
 pass  
  
class ElektrikliAraba(Araba):  
 def \_\_init\_\_(self, marka, model, yil, batarya\_durumu):  
 super().\_\_init\_\_(marka, model, yil)  
 self.batarya\_durumu = batarya\_durumu  
  
 def bakim(self):  
 return f"{self.marka} {self.model} elektrikli araba bakımı: Pil sağlığı kontrol ediliyor. Batarya Durumu: {self.batarya\_durumu}"  
  
class BenzinliAraba(Araba):  
 def \_\_init\_\_(self, marka, model, yil, yakit\_durumu):  
 super().\_\_init\_\_(marka, model, yil)  
 self.yakit\_durumu = yakit\_durumu  
  
 def bakim(self):  
 return f"{self.marka} {self.model} benzinli araba bakımı: Yağ değiştiriliyor. Yakıt Durumu: {self.yakit\_durumu}"  
class Kiralama:  
 def \_\_init\_\_(self, musteri, araba, kiralama\_tipi, sure):  
 self.musteri = musteri  
 self.araba = araba  
 self.kiralama\_tipi = kiralama\_tipi  
 self.sure = sure  
 self.baslangic\_tarihi = datetime.datetime.now()  
  
 def hesapla\_ucret(self):  
 if self.kiralama\_tipi == "günlük":  
 return self.sure \* 1000  
 elif self.kiralama\_tipi == "saatlik":  
 return self.sure \* 500  
 else:  
 print("Geçersiz kiralama tipi")  
 return -1  
  
 def kiralama\_bitir(self):  
 bitis\_tarihi = datetime.datetime.now()  
 kiralama\_suresi = (bitis\_tarihi - self.baslangic\_tarihi).total\_seconds() / 3600  
  
 self.araba.set\_kilometre(self.araba.get\_kilometre() + kiralama\_suresi \* 10) # 10 km/saat varsayalım  
  
class AracKiralamaUygulamasi:  
 def \_\_init\_\_(self, root):  
 self.root = root  
 self.root.title("Araç Kiralama Uygulaması")  
  
 self.create\_database\_connection()  
  
 self.create\_main\_frame()  
  
 def create\_database\_connection(self):  
 self.baglanti = sqlite3.connect("rent-a.db")  
 self.cursor = self.baglanti.cursor()  
  
 def create\_main\_frame(self):  
 self.main\_frame = ttk.Frame(self.root, padding="20")  
 self.main\_frame.grid(row=0, column=0, sticky=(tk.W, tk.E, tk.N, tk.S))  
  
 self.label\_welcome = ttk.Label(self.main\_frame, text="Araç Kiralama Uygulamasına Hoş Geldiniz!", font=('Arial', 16, 'bold', "italic"))  
 self.label\_welcome.grid(row=0, column=0, columnspan=3, pady=20)  
  
 self.label\_instruction = ttk.Label(self.main\_frame, text="Adınız Soyadınız:", font=('Arial', 12))  
 self.label\_instruction.grid(row=1, column=0, columnspan=3, pady=10)  
  
 self.entry\_ad\_soyad = ttk.Entry(self.main\_frame, font=('Arial', 12))  
 self.entry\_ad\_soyad.grid(row=2, column=0, columnspan=3, pady=10)  
  
 # Giriş yap butonu ve stili  
 self.button\_giris = tk.Button(self.main\_frame, text="Giriş Yap", command=self.musteri\_giris,  
 width=10, height=1, bg="purple", fg="white", font=("Arial", 12))  
 self.button\_giris.grid(row=3, column=0, columnspan=3, pady=20)  
  
 def musteri\_giris(self):  
 ad\_soyad = self.entry\_ad\_soyad.get()  
  
 if not ad\_soyad:  
 messagebox.showwarning("Uyarı", "Lütfen adınızı ve soyadınızı girin.")  
 return  
  
 musteri\_id = self.add\_musteri(ad\_soyad)  
 if musteri\_id:  
 self.musteri\_id = musteri\_id  
 self.label\_welcome.config(text=f"Hoş Geldin, {ad\_soyad}!")  
 self.label\_instruction.grid\_remove()  
 self.entry\_ad\_soyad.grid\_remove()  
 self.button\_giris.grid\_remove()  
  
 self.show\_arac\_list()  
  
 def add\_musteri(self, ad\_soyad):  
 self.cursor.execute("INSERT INTO musteri (ad\_soyad) VALUES (?)", (ad\_soyad,))  
 self.baglanti.commit()  
 return self.cursor.lastrowid  
  
 def show\_arac\_list(self):  
 araclar = self.get\_arac\_list()  
  
 self.tree\_araclar = ttk.Treeview(self.main\_frame, columns=("id", "Marka", "Model", "Yıl", "Tip", "Durum"), show="headings")  
 self.tree\_araclar.heading("id", text="id")  
 self.tree\_araclar.heading("Marka", text="Marka")  
 self.tree\_araclar.heading("Model", text="Model")  
 self.tree\_araclar.heading("Yıl", text="Yıl")  
 self.tree\_araclar.heading("Tip", text="Tip")  
 self.tree\_araclar.heading("Durum", text="Durum")  
  
 for arac in araclar:  
 self.tree\_araclar.insert("", "end", values=arac)  
  
 self.tree\_araclar.grid(row=1, column=0, columnspan=3, sticky=(tk.W, tk.E))  
  
 self.button\_kirala = ttk.Button(self.main\_frame, text="Araç Kirala", command=self.arac\_kirala)  
 self.button\_kirala.grid(row=2, column=0, columnspan=3, pady=10, sticky=(tk.W, tk.E))  
  
 def get\_arac\_list(self):  
 self.cursor.execute(  
 "SELECT arabalar.id, marka, model, yil, tip, durumu FROM arabalar WHERE durumu IN ('müsait', 'kiralandı')")  
 return self.cursor.fetchall()  
  
 def arac\_kirala(self):  
 selected\_item = self.tree\_araclar.selection()  
 if selected\_item:  
 arac\_id = self.tree\_araclar.item(selected\_item, "values")[0]  
 arac\_durumu = self.tree\_araclar.item(selected\_item, "values")[5] # Assuming status column is the sixth index  
  
 if arac\_durumu == 'kiralandı':  
 messagebox.showwarning("Uyarı", "Bu araç zaten kiralanmış!")  
 return  
  
 kiralama\_tipi = self.show\_kiralama\_tipi\_dialog()  
 if kiralama\_tipi:  
 sure = self.show\_kiralama\_sure\_dialog(kiralama\_tipi)  
  
 if sure:  
 arac = self.get\_arac\_by\_id(arac\_id)  
  
 # Aracın bakım durumu GUI'de gösteriliyor  
 self.show\_arac\_bakim\_durumu(arac)  
  
 # Aracın durumunu güncelle  
 self.update\_arac\_durumu(arac\_id)  
  
 # Kiralama bilgilerini veritabanına ekle  
 kiralama = Kiralama(self.musteri\_id, arac, kiralama\_tipi, sure)  
 self.add\_kiralama\_bilgisi(kiralama)  
  
 ucret = kiralama.hesapla\_ucret()  
 if ucret != -1:  
 messagebox.showinfo("Başarılı", f"Araç kiralandı! Ücret: {ucret} TL")  
 kiralama.kiralama\_bitir()  
  
 self.root.destroy()  
  
 def update\_arac\_durumu(self, arac\_id):  
 self.cursor.execute("UPDATE arabalar SET durumu = 'kiralandı' WHERE id = ?", (arac\_id,))  
 self.baglanti.commit()  
  
 def add\_kiralama\_bilgisi(self, kiralama):  
 ucret = kiralama.hesapla\_ucret() # Calculate the rental fee  
  
 self.cursor.execute(  
 "INSERT INTO kiralama (musteri\_id, araba\_id, kiralama\_tipi, sure, ucret, durumu) VALUES ( ?, ?, ?, ?, ?, ?)",  
 (kiralama.musteri, kiralama.araba.id, kiralama.kiralama\_tipi, kiralama.sure, ucret, 'kiralandı'))  
 self.baglanti.commit()  
  
 def show\_arac\_bakim\_durumu(self, arac):  
 bakim\_durumu = arac.bakim()  
 messagebox.showinfo("Bakım Durumu", bakim\_durumu)  
  
 def show\_kiralama\_tipi\_dialog(self):  
 while True:  
 kiralama\_tipi = simpledialog.askstring("Kiralama Tipi", "Kiralama Tipini Seçiniz (günlük/saatlik)")  
  
 if kiralama\_tipi is None:  
 return None # Eğer kullanıcı iptal ederse  
  
 kiralama\_tipi\_lower = kiralama\_tipi.lower()  
 if kiralama\_tipi\_lower not in ["günlük", "saatlik"]:  
 messagebox.showwarning("Uyarı", "Lütfen 'günlük' veya 'saatlik' olarak girin.")  
 else:  
 return kiralama\_tipi\_lower  
  
 def show\_kiralama\_sure\_dialog(self, kiralama\_tip):  
 while True:  
 if kiralama\_tip == "günlük":  
 message = "Günlük Kiralama Süresini Giriniz (gün)"  
 elif kiralama\_tip == "saatlik":  
 message = "Saatlik Kiralama Süresini Giriniz (saat)"  
  
 kiralama\_suresi = simpledialog.askinteger("Kiralama Süresi", message)  
  
 if kiralama\_suresi is None:  
 return None # Eğer kullanıcı iptal ederse  
  
 if kiralama\_suresi <= 0:  
 messagebox.showwarning("Uyarı", "Lütfen pozitif bir değer girin.")  
 else:  
 return kiralama\_suresi  
  
 def get\_arac\_by\_id(self, arac\_id):  
 self.cursor.execute("SELECT \* FROM arabalar WHERE id = ?", (arac\_id,))  
 arac\_data = self.cursor.fetchone()  
  
 if arac\_data:  
 arac\_id, marka, model, yil, tip, durumu,kilometre = arac\_data  
 if tip == 'Elektrikli':  
 elektrikli\_arac = ElektrikliAraba(marka, model, yil, batarya\_durumu="Yüksek")  
 elektrikli\_arac.id = arac\_id # Set the id attribute  
 return elektrikli\_arac  
 elif tip == 'Benzinli':  
 benzinli\_arac = BenzinliAraba(marka, model, yil, yakit\_durumu="Doluluk Oranı: %100")  
 benzinli\_arac.id = arac\_id # Set the id attribute  
 return benzinli\_arac  
 else:  
 return None  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 root = tk.Tk()  
 app = AracKiralamaUygulamasi(root)  
 root.mainloop()

import sqlite3  
  
baglanti = sqlite3.connect("rent-a.db")  
cursor = baglanti.cursor()  
  
cursor.execute('''  
 CREATE TABLE IF NOT EXISTS musteri (  
 id INTEGER PRIMARY KEY AUTOINCREMENT,  
 ad\_soyad TEXT  
 )  
''')  
  
# Araba tablosunu oluştur  
cursor.execute('''  
 CREATE TABLE IF NOT EXISTS arabalar (  
 id INTEGER PRIMARY KEY AUTOINCREMENT,  
 marka TEXT,  
 model TEXT,  
 yil INTEGER,  
 tip TEXT,  
 durumu TEXT DEFAULT 'müsait',  
 kilometre INTEGER DEFAULT 0  
 )  
''')  
  
# Kiralama tablosunu oluştur  
cursor.execute('''  
 CREATE TABLE IF NOT EXISTS kiralama (  
 musteri\_id INTEGER,  
 araba\_id INTEGER,  
 kiralama\_tipi TEXT,  
 baslangic\_tarihi DATETIME,   
 bitis\_tarihi DATETIME,   
 baslangic\_km INTEGER,   
 bitis\_km INTEGER,   
 sure INTEGER,  
 ucret INTEGER,  
 durumu TEXT DEFAULT 'kiralandı',  
 FOREIGN KEY (araba\_id) REFERENCES arabalar (id),  
 FOREIGN KEY (musteri\_id) REFERENCES musteri(id)  
   
 )  
''')  
  
  
  
# Elektrikli Arabalar  
cursor.execute("INSERT INTO arabalar (marka, model, yil, tip) VALUES ('Tesla', 'Model S', 2023, 'Elektrikli')")  
cursor.execute("INSERT INTO arabalar (marka, model, yil, tip) VALUES ('Porsche', 'Taycan', 2023, 'Elektrikli')")  
cursor.execute("INSERT INTO arabalar (marka, model, yil, tip) VALUES ('Audi', 'e-tron', 2023, 'Elektrikli')")  
cursor.execute("INSERT INTO arabalar (marka, model, yil, tip) VALUES ('Mercedes-Benz', 'EQC', 2023, 'Elektrikli')")  
cursor.execute("INSERT INTO arabalar (marka, model, yil, tip) VALUES ('Jaguar', 'I-PACE', 2023, 'Elektrikli')")  
  
# Benzinli Arabalar  
cursor.execute("INSERT INTO arabalar (marka, model, yil, tip) VALUES ('Mercedes-Benz', 'S-Class', 2023, 'Benzinli')")  
cursor.execute("INSERT INTO arabalar (marka, model, yil, tip) VALUES ('BMW', '7 Series', 2023, 'Benzinli')")  
cursor.execute("INSERT INTO arabalar (marka, model, yil, tip) VALUES ('Lexus', 'LS', 2023, 'Benzinli')")  
cursor.execute("INSERT INTO arabalar (marka, model, yil, tip) VALUES ('Audi', 'A8', 2023, 'Benzinli')")  
cursor.execute("INSERT INTO arabalar (marka, model, yil, tip) VALUES ('Porsche', 'Panamera', 2023, 'Benzinli')")  
  
baglanti.commit()  
# Veritabanı bağlantısını kapat  
baglanti.close()

17.25 18.12.20223 başlangıç bitiş kilometre eklicem

import sqlite3  
  
baglanti = sqlite3.connect("rent-a.db")  
cursor = baglanti.cursor()  
  
cursor.execute('''  
 CREATE TABLE IF NOT EXISTS musteri (  
 id INTEGER PRIMARY KEY AUTOINCREMENT,  
 ad\_soyad TEXT  
 )  
''')  
  
# Araba tablosunu oluştur  
cursor.execute('''  
 CREATE TABLE IF NOT EXISTS arabalar (  
 id INTEGER PRIMARY KEY AUTOINCREMENT,  
 marka TEXT,  
 model TEXT,  
 yil INTEGER,  
 tip TEXT,  
 durumu TEXT DEFAULT 'müsait'  
 )  
''')  
  
# Kiralama tablosunu oluştur  
cursor.execute('''  
 CREATE TABLE IF NOT EXISTS kiralama (  
 musteri\_id INTEGER,  
 araba\_id INTEGER,  
 kiralama\_tipi TEXT,  
 sure INTEGER,  
 ucret INTEGER,  
 durumu TEXT DEFAULT 'kiralandı',  
 FOREIGN KEY (araba\_id) REFERENCES arabalar (id),  
 FOREIGN KEY (musteri\_id) REFERENCES musteri(id)  
   
 )  
''')  
  
  
  
# Elektrikli Arabalar  
cursor.execute("INSERT INTO arabalar (marka, model, yil, tip) VALUES ('Tesla', 'Model S', 2023, 'Elektrikli')")  
cursor.execute("INSERT INTO arabalar (marka, model, yil, tip) VALUES ('Porsche', 'Taycan', 2023, 'Elektrikli')")  
cursor.execute("INSERT INTO arabalar (marka, model, yil, tip) VALUES ('Audi', 'e-tron', 2023, 'Elektrikli')")  
cursor.execute("INSERT INTO arabalar (marka, model, yil, tip) VALUES ('Mercedes-Benz', 'EQC', 2023, 'Elektrikli')")  
cursor.execute("INSERT INTO arabalar (marka, model, yil, tip) VALUES ('Jaguar', 'I-PACE', 2023, 'Elektrikli')")  
  
# Benzinli Arabalar  
cursor.execute("INSERT INTO arabalar (marka, model, yil, tip) VALUES ('Mercedes-Benz', 'S-Class', 2023, 'Benzinli')")  
cursor.execute("INSERT INTO arabalar (marka, model, yil, tip) VALUES ('BMW', '7 Series', 2023, 'Benzinli')")  
cursor.execute("INSERT INTO arabalar (marka, model, yil, tip) VALUES ('Lexus', 'LS', 2023, 'Benzinli')")  
cursor.execute("INSERT INTO arabalar (marka, model, yil, tip) VALUES ('Audi', 'A8', 2023, 'Benzinli')")  
cursor.execute("INSERT INTO arabalar (marka, model, yil, tip) VALUES ('Porsche', 'Panamera', 2023, 'Benzinli')")  
  
baglanti.commit()  
# Veritabanı bağlantısını kapat  
baglanti.close()

import tkinter as tk  
from tkinter import ttk, simpledialog, messagebox  
import sqlite3  
import datetime  
from abc import ABC, abstractmethod  
class Araba(ABC):  
 def \_\_init\_\_(self, marka, model, yil):  
 self.id = None  
 self.marka = marka  
 self.model = model  
 self.\_yil = yil # Protected özellik  
 self.\_\_kilometre = 0 # Private özellik  
  
 def get\_yil(self):  
 return self.\_yil  
  
 def set\_kilometre(self, km):  
 if km >= 0:  
 self.\_\_kilometre = km  
 else:  
 print("Geçersiz kilometre değeri.")  
  
 def get\_kilometre(self):  
 return self.\_\_kilometre  
  
 @abstractmethod  
 def bakim(self):  
 pass  
  
class ElektrikliAraba(Araba):  
 def \_\_init\_\_(self, marka, model, yil, batarya\_durumu):  
 super().\_\_init\_\_(marka, model, yil)  
 self.batarya\_durumu = batarya\_durumu  
  
 def bakim(self):  
 return f"{self.marka} {self.model} elektrikli araba bakımı: Pil sağlığı kontrol ediliyor. Batarya Durumu: {self.batarya\_durumu}"  
  
class BenzinliAraba(Araba):  
 def \_\_init\_\_(self, marka, model, yil, yakit\_durumu):  
 super().\_\_init\_\_(marka, model, yil)  
 self.yakit\_durumu = yakit\_durumu  
  
 def bakim(self):  
 return f"{self.marka} {self.model} benzinli araba bakımı: Yağ değiştiriliyor. Yakıt Durumu: {self.yakit\_durumu}"  
class Kiralama:  
 def \_\_init\_\_(self, musteri, araba, kiralama\_tipi, sure):  
 self.musteri = musteri  
 self.araba = araba  
 self.kiralama\_tipi = kiralama\_tipi  
 self.sure = sure  
 self.baslangic\_tarihi = datetime.datetime.now()  
  
 def hesapla\_ucret(self):  
 if self.kiralama\_tipi == "günlük":  
 return self.sure \* 1000  
 elif self.kiralama\_tipi == "saatlik":  
 return self.sure \* 500  
 else:  
 print("Geçersiz kiralama tipi")  
 return -1  
  
 def kiralama\_bitir(self):  
 bitis\_tarihi = datetime.datetime.now()  
 kiralama\_suresi = (bitis\_tarihi - self.baslangic\_tarihi).total\_seconds() / 3600  
  
 self.araba.set\_kilometre(self.araba.get\_kilometre() + kiralama\_suresi \* 10) # 10 km/saat varsayalım  
  
class AracKiralamaUygulamasi:  
 def \_\_init\_\_(self, root):  
 self.root = root  
 self.root.title("Araç Kiralama Uygulaması")  
  
 self.create\_database\_connection()  
  
 self.create\_main\_frame()  
  
 def create\_database\_connection(self):  
 self.baglanti = sqlite3.connect("rent-a.db")  
 self.cursor = self.baglanti.cursor()  
  
 def create\_main\_frame(self):  
 self.main\_frame = ttk.Frame(self.root, padding="20")  
 self.main\_frame.grid(row=0, column=0, sticky=(tk.W, tk.E, tk.N, tk.S))  
  
 self.label\_welcome = ttk.Label(self.main\_frame, text="Araç Kiralama Uygulamasına Hoş Geldiniz!", font=('Arial', 16, 'bold', "italic"))  
 self.label\_welcome.grid(row=0, column=0, columnspan=3, pady=20)  
  
 self.label\_instruction = ttk.Label(self.main\_frame, text="Adınız Soyadınız:", font=('Arial', 12))  
 self.label\_instruction.grid(row=1, column=0, columnspan=3, pady=10)  
  
 self.entry\_ad\_soyad = ttk.Entry(self.main\_frame, font=('Arial', 12))  
 self.entry\_ad\_soyad.grid(row=2, column=0, columnspan=3, pady=10)  
  
 # Giriş yap butonu ve stili  
 self.button\_giris = tk.Button(self.main\_frame, text="Giriş Yap", command=self.musteri\_giris,  
 width=10, height=1, bg="purple", fg="white", font=("Arial", 12))  
 self.button\_giris.grid(row=3, column=0, columnspan=3, pady=20)  
  
 def musteri\_giris(self):  
 ad\_soyad = self.entry\_ad\_soyad.get()  
  
 if not ad\_soyad:  
 messagebox.showwarning("Uyarı", "Lütfen adınızı ve soyadınızı girin.")  
 return  
  
 musteri\_id = self.add\_musteri(ad\_soyad)  
 if musteri\_id:  
 self.musteri\_id = musteri\_id  
 self.label\_welcome.config(text=f"Hoş Geldin, {ad\_soyad}!")  
 self.label\_instruction.grid\_remove()  
 self.entry\_ad\_soyad.grid\_remove()  
 self.button\_giris.grid\_remove()  
  
 self.show\_arac\_list()  
  
 def add\_musteri(self, ad\_soyad):  
 self.cursor.execute("INSERT INTO musteri (ad\_soyad) VALUES (?)", (ad\_soyad,))  
 self.baglanti.commit()  
 return self.cursor.lastrowid  
  
 def show\_arac\_list(self):  
 araclar = self.get\_arac\_list()  
  
 self.tree\_araclar = ttk.Treeview(self.main\_frame, columns=("Marka", "Model", "Yıl", "Tip", "Durum"), show="headings")  
 self.tree\_araclar.heading("Marka", text="Marka")  
 self.tree\_araclar.heading("Model", text="Model")  
 self.tree\_araclar.heading("Yıl", text="Yıl")  
 self.tree\_araclar.heading("Tip", text="Tip")  
 self.tree\_araclar.heading("Durum", text="Durum")  
  
 for arac in araclar:  
 self.tree\_araclar.insert("", "end", values=arac)  
  
 self.tree\_araclar.grid(row=1, column=0, columnspan=3, sticky=(tk.W, tk.E))  
  
 self.button\_kirala = ttk.Button(self.main\_frame, text="Araç Kirala", command=self.arac\_kirala)  
 self.button\_kirala.grid(row=2, column=0, columnspan=3, pady=10, sticky=(tk.W, tk.E))  
  
 def get\_arac\_list(self):  
 self.cursor.execute("SELECT arabalar.id, marka, model, yil, tip, durumu FROM arabalar WHERE durumu = 'müsait'")  
 return self.cursor.fetchall()  
  
 def arac\_kirala(self):  
 selected\_item = self.tree\_araclar.selection()  
 if selected\_item:  
 arac\_id = self.tree\_araclar.item(selected\_item, "values")[0]  
  
 kiralama\_tipi = self.show\_kiralama\_tipi\_dialog()  
 if kiralama\_tipi:  
 sure = self.show\_kiralama\_sure\_dialog(kiralama\_tipi)  
  
 if sure:  
 arac = self.get\_arac\_by\_id(arac\_id)  
  
 # Aracın bakım durumu GUI'de gösteriliyor  
 self.show\_arac\_bakim\_durumu(arac)  
  
 # Aracın durumunu güncelle  
 self.update\_arac\_durumu(arac\_id)  
  
 # Kiralama bilgilerini veritabanına ekle  
 kiralama = Kiralama(self.musteri\_id, arac, kiralama\_tipi, sure)  
 self.add\_kiralama\_bilgisi(kiralama)  
  
 ucret = kiralama.hesapla\_ucret()  
 if ucret != -1:  
 messagebox.showinfo("Başarılı", f"Araç kiralandı! Ücret: {ucret} TL")  
 kiralama.kiralama\_bitir()  
  
 def update\_arac\_durumu(self, arac\_id):  
 self.cursor.execute("UPDATE arabalar SET durumu = 'kiralandı' WHERE id = ?", (arac\_id,))  
 self.baglanti.commit()  
  
 def add\_kiralama\_bilgisi(self, kiralama):  
 ucret = kiralama.hesapla\_ucret() # Calculate the rental fee  
  
 self.cursor.execute(  
 "INSERT INTO kiralama (musteri\_id, araba\_id, kiralama\_tipi, sure, ucret, durumu) VALUES ( ?, ?, ?, ?, ?, ?)",  
 (kiralama.musteri, kiralama.araba.id, kiralama.kiralama\_tipi, kiralama.sure, ucret, 'kiralandı'))  
 self.baglanti.commit()  
  
 def show\_arac\_bakim\_durumu(self, arac):  
 bakim\_durumu = arac.bakim()  
 messagebox.showinfo("Bakım Durumu", bakim\_durumu)  
  
 def show\_kiralama\_tipi\_dialog(self):  
 while True:  
 kiralama\_tipi = simpledialog.askstring("Kiralama Tipi", "Kiralama Tipini Seçiniz (günlük/saatlik)")  
  
 if kiralama\_tipi is None:  
 return None # Eğer kullanıcı iptal ederse  
  
 kiralama\_tipi\_lower = kiralama\_tipi.lower()  
 if kiralama\_tipi\_lower not in ["günlük", "saatlik"]:  
 messagebox.showwarning("Uyarı", "Lütfen 'günlük' veya 'saatlik' olarak girin.")  
 else:  
 return kiralama\_tipi\_lower  
  
 def show\_kiralama\_sure\_dialog(self, kiralama\_tip):  
 while True:  
 if kiralama\_tip == "günlük":  
 message = "Günlük Kiralama Süresini Giriniz (gün)"  
 elif kiralama\_tip == "saatlik":  
 message = "Saatlik Kiralama Süresini Giriniz (saat)"  
  
 kiralama\_suresi = simpledialog.askinteger("Kiralama Süresi", message)  
  
 if kiralama\_suresi is None:  
 return None # Eğer kullanıcı iptal ederse  
  
 if kiralama\_suresi <= 0:  
 messagebox.showwarning("Uyarı", "Lütfen pozitif bir değer girin.")  
 else:  
 return kiralama\_suresi  
  
 def get\_arac\_by\_id(self, arac\_id):  
 self.cursor.execute("SELECT \* FROM arabalar WHERE id = ?", (arac\_id,))  
 arac\_data = self.cursor.fetchone()  
  
 if arac\_data:  
 arac\_id, marka, model, yil, tip, durumu = arac\_data  
 if tip == 'Elektrikli':  
 elektrikli\_arac = ElektrikliAraba(marka, model, yil, batarya\_durumu="Yüksek")  
 elektrikli\_arac.id = arac\_id # Set the id attribute  
 return elektrikli\_arac  
 elif tip == 'Benzinli':  
 benzinli\_arac = BenzinliAraba(marka, model, yil, yakit\_durumu="Doluluk Oranı: %100")  
 benzinli\_arac.id = arac\_id # Set the id attribute  
 return benzinli\_arac  
 else:  
 return None  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 root = tk.Tk()  
 app = AracKiralamaUygulamasi(root)  
 root.mainloop()

18.26 17.12.2023

import tkinter as tk  
from tkinter import ttk, simpledialog, messagebox  
import sqlite3  
import datetime  
from abc import ABC, abstractmethod  
class Araba(ABC):  
 def \_\_init\_\_(self, marka, model, yil):  
 self.id = None  
 self.marka = marka  
 self.model = model  
 self.\_yil = yil # Protected özellik  
 self.\_\_kilometre = 0 # Private özellik  
  
 def get\_yil(self):  
 return self.\_yil  
  
 def set\_kilometre(self, km):  
 if km >= 0:  
 self.\_\_kilometre = km  
 else:  
 print("Geçersiz kilometre değeri.")  
  
 def get\_kilometre(self):  
 return self.\_\_kilometre  
  
 @abstractmethod  
 def bakim(self):  
 pass  
  
class ElektrikliAraba(Araba):  
 def \_\_init\_\_(self, marka, model, yil, batarya\_durumu):  
 super().\_\_init\_\_(marka, model, yil)  
 self.batarya\_durumu = batarya\_durumu  
  
 def bakim(self):  
 return f"{self.marka} {self.model} elektrikli araba bakımı: Pil sağlığı kontrol ediliyor. Batarya Durumu: {self.batarya\_durumu}"  
  
class BenzinliAraba(Araba):  
 def \_\_init\_\_(self, marka, model, yil, yakit\_durumu):  
 super().\_\_init\_\_(marka, model, yil)  
 self.yakit\_durumu = yakit\_durumu  
  
 def bakim(self):  
 return f"{self.marka} {self.model} benzinli araba bakımı: Yağ değiştiriliyor. Yakıt Durumu: {self.yakit\_durumu}"  
class Kiralama:  
 def \_\_init\_\_(self, musteri, araba, kiralama\_tipi, sure):  
 self.musteri = musteri  
 self.araba = araba  
 self.kiralama\_tipi = kiralama\_tipi  
 self.sure = sure  
 self.baslangic\_tarihi = datetime.datetime.now()  
  
 def hesapla\_ucret(self):  
 if self.kiralama\_tipi == "günlük":  
 return self.sure \* 100  
 elif self.kiralama\_tipi == "saatlik":  
 return self.sure \* 20  
 else:  
 print("Geçersiz kiralama tipi")  
 return -1  
  
 # return ucret  
  
 def kiralama\_bitir(self):  
 bitis\_tarihi = datetime.datetime.now()  
 kiralama\_suresi = (bitis\_tarihi - self.baslangic\_tarihi).total\_seconds() / 3600  
  
 self.araba.set\_kilometre(self.araba.get\_kilometre() + kiralama\_suresi \* 10) # 10 km/saat varsayalım  
  
class AracKiralamaUygulamasi:  
 def \_\_init\_\_(self, root):  
 self.root = root  
 self.root.title("Araç Kiralama Uygulaması")  
  
 self.create\_database\_connection()  
  
 self.create\_main\_frame()  
  
 def create\_database\_connection(self):  
 self.baglanti = sqlite3.connect("rent-a.db")  
 self.cursor = self.baglanti.cursor()  
  
 def create\_main\_frame(self):  
 self.main\_frame = ttk.Frame(self.root, padding="20")  
 self.main\_frame.grid(row=0, column=0, sticky=(tk.W, tk.E, tk.N, tk.S))  
  
 self.label\_welcome = ttk.Label(self.main\_frame, text="Araç Kiralama Uygulamasına Hoş Geldiniz!", font=('Arial', 16, 'bold', "italic"))  
 self.label\_welcome.grid(row=0, column=0, columnspan=3, pady=20)  
  
 self.label\_instruction = ttk.Label(self.main\_frame, text="Adınız Soyadınız:", font=('Arial', 12))  
 self.label\_instruction.grid(row=1, column=0, columnspan=3, pady=10)  
  
 self.entry\_ad\_soyad = ttk.Entry(self.main\_frame, font=('Arial', 12))  
 self.entry\_ad\_soyad.grid(row=2, column=0, columnspan=3, pady=10)  
  
 # Giriş yap butonu ve stili  
 self.button\_giris = tk.Button(self.main\_frame, text="Giriş Yap", command=self.musteri\_giris,  
 width=10, height=1, bg="purple", fg="white", font=("Arial", 12))  
 self.button\_giris.grid(row=3, column=0, columnspan=3, pady=20)  
  
 def musteri\_giris(self):  
 ad\_soyad = self.entry\_ad\_soyad.get()  
  
 if not ad\_soyad:  
 messagebox.showwarning("Uyarı", "Lütfen adınızı ve soyadınızı girin.")  
 return  
  
 musteri\_id = self.add\_musteri(ad\_soyad)  
 if musteri\_id:  
 self.musteri\_id = musteri\_id  
 self.label\_welcome.config(text=f"Hoş Geldin, {ad\_soyad}!")  
 self.label\_instruction.grid\_remove()  
 self.entry\_ad\_soyad.grid\_remove()  
 self.button\_giris.grid\_remove()  
  
 self.show\_arac\_list()  
  
 def add\_musteri(self, ad\_soyad):  
 self.cursor.execute("INSERT INTO musteri (ad\_soyad) VALUES (?)", (ad\_soyad,))  
 self.baglanti.commit()  
 return self.cursor.lastrowid  
  
 def show\_arac\_list(self):  
 araclar = self.get\_arac\_list()  
  
 self.tree\_araclar = ttk.Treeview(self.main\_frame, columns=("Marka", "Model", "Yıl", "Tip", "Durum"), show="headings")  
 self.tree\_araclar.heading("Marka", text="Marka")  
 self.tree\_araclar.heading("Model", text="Model")  
 self.tree\_araclar.heading("Yıl", text="Yıl")  
 self.tree\_araclar.heading("Tip", text="Tip")  
 self.tree\_araclar.heading("Durum", text="Durum")  
  
 for arac in araclar:  
 self.tree\_araclar.insert("", "end", values=arac)  
  
 self.tree\_araclar.grid(row=1, column=0, columnspan=3, sticky=(tk.W, tk.E))  
  
 self.button\_kirala = ttk.Button(self.main\_frame, text="Araç Kirala", command=self.arac\_kirala)  
 self.button\_kirala.grid(row=2, column=0, columnspan=3, pady=10, sticky=(tk.W, tk.E))  
  
 def get\_arac\_list(self):  
 self.cursor.execute("SELECT arabalar.id, marka, model, yil, tip, durumu FROM arabalar WHERE durumu = 'müsait'")  
 return self.cursor.fetchall()  
  
 def arac\_kirala(self):  
 selected\_item = self.tree\_araclar.selection()  
 if selected\_item:  
 arac\_id = self.tree\_araclar.item(selected\_item, "values")[0]  
  
 kiralama\_tipi = self.show\_kiralama\_tipi\_dialog()  
 if kiralama\_tipi:  
 sure = self.show\_kiralama\_sure\_dialog(kiralama\_tipi)  
  
 if sure:  
 arac = self.get\_arac\_by\_id(arac\_id)  
  
 # Aracın bakım durumu GUI'de gösteriliyor  
 self.show\_arac\_bakim\_durumu(arac)  
  
 # Aracın durumunu güncelle  
 self.update\_arac\_durumu(arac\_id)  
  
 # Kiralama bilgilerini veritabanına ekle  
 kiralama = Kiralama(self.musteri\_id, arac, kiralama\_tipi, sure)  
 self.add\_kiralama\_bilgisi(kiralama)  
  
 ucret = kiralama.hesapla\_ucret()  
 if ucret != -1:  
 messagebox.showinfo("Başarılı", f"Araç kiralandı! Ücret: {ucret} TL")  
 kiralama.kiralama\_bitir()  
  
 def update\_arac\_durumu(self, arac\_id):  
 self.cursor.execute("UPDATE arabalar SET durumu = 'kiralandı' WHERE id = ?", (arac\_id,))  
 self.baglanti.commit()  
  
 def add\_kiralama\_bilgisi(self, kiralama):  
 ucret = kiralama.hesapla\_ucret() # Calculate the rental fee  
  
 self.cursor.execute(  
 "INSERT INTO kiralama (musteri\_id, araba\_id, kiralama\_tipi, sure, ucret, durumu) VALUES ( ?, ?, ?, ?, ?, ?)",  
 (kiralama.musteri, kiralama.araba.id, kiralama.kiralama\_tipi, kiralama.sure, ucret, 'kiralandı'))  
 self.baglanti.commit()  
  
 def show\_arac\_bakim\_durumu(self, arac):  
 bakim\_durumu = arac.bakim()  
 messagebox.showinfo("Bakım Durumu", bakim\_durumu)  
  
 def show\_kiralama\_tipi\_dialog(self):  
 while True:  
 kiralama\_tipi = simpledialog.askstring("Kiralama Tipi", "Kiralama Tipini Seçiniz (günlük/saatlik)")  
  
 if kiralama\_tipi is None:  
 return None # Eğer kullanıcı iptal ederse  
  
 kiralama\_tipi\_lower = kiralama\_tipi.lower()  
 if kiralama\_tipi\_lower not in ["günlük", "saatlik"]:  
 messagebox.showwarning("Uyarı", "Lütfen 'günlük' veya 'saatlik' olarak girin.")  
 else:  
 return kiralama\_tipi\_lower  
  
 def show\_kiralama\_sure\_dialog(self, kiralama\_tip):  
 while True:  
 if kiralama\_tip == "günlük":  
 message = "Günlük Kiralama Süresini Giriniz (gün)"  
 elif kiralama\_tip == "saatlik":  
 message = "Saatlik Kiralama Süresini Giriniz (saat)"  
  
 kiralama\_suresi = simpledialog.askinteger("Kiralama Süresi", message)  
  
 if kiralama\_suresi is None:  
 return None # Eğer kullanıcı iptal ederse  
  
 if kiralama\_suresi <= 0:  
 messagebox.showwarning("Uyarı", "Lütfen pozitif bir değer girin.")  
 else:  
 return kiralama\_suresi  
  
 def get\_arac\_by\_id(self, arac\_id):  
 self.cursor.execute("SELECT \* FROM arabalar WHERE id = ?", (arac\_id,))  
 arac\_data = self.cursor.fetchone()  
  
 if arac\_data:  
 arac\_id, marka, model, yil, tip, durumu = arac\_data  
 if tip == 'Elektrikli':  
 elektrikli\_arac = ElektrikliAraba(marka, model, yil, batarya\_durumu="Yüksek")  
 elektrikli\_arac.id = arac\_id # Set the id attribute  
 return elektrikli\_arac  
 elif tip == 'Benzinli':  
 benzinli\_arac = BenzinliAraba(marka, model, yil, yakit\_durumu="Doluluk Oranı: %100")  
 benzinli\_arac.id = arac\_id # Set the id attribute  
 return benzinli\_arac  
 else:  
 return None  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 root = tk.Tk()  
 app = AracKiralamaUygulamasi(root)  
 root.mainloop()

import sqlite3  
  
baglanti = sqlite3.connect("rent-a.db")  
cursor = baglanti.cursor()  
  
cursor.execute('''  
 CREATE TABLE IF NOT EXISTS musteri (  
 id INTEGER PRIMARY KEY AUTOINCREMENT,  
 ad\_soyad TEXT  
 )  
''')  
  
# Araba tablosunu oluştur  
cursor.execute('''  
 CREATE TABLE IF NOT EXISTS arabalar (  
 id INTEGER PRIMARY KEY AUTOINCREMENT,  
 marka TEXT,  
 model TEXT,  
 yil INTEGER,  
 tip TEXT,  
 durumu TEXT DEFAULT 'müsait'  
 )  
''')  
  
# Kiralama tablosunu oluştur  
cursor.execute('''  
 CREATE TABLE IF NOT EXISTS kiralama (  
 musteri\_id TEXT,  
 araba\_id INTEGER,  
 kiralama\_tipi TEXT,  
 sure INTEGER,  
 ucret INTEGER,  
 durumu TEXT DEFAULT 'kiralandı',  
 FOREIGN KEY (araba\_id) REFERENCES arabalar (id),  
 FOREIGN KEY (musteri\_id) REFERENCES musteri(id)  
   
 )  
''')  
  
  
  
# Elektrikli Arabalar  
cursor.execute("INSERT INTO arabalar (marka, model, yil, tip) VALUES ('Tesla', 'Model S', 2023, 'Elektrikli')")  
cursor.execute("INSERT INTO arabalar (marka, model, yil, tip) VALUES ('Porsche', 'Taycan', 2023, 'Elektrikli')")  
cursor.execute("INSERT INTO arabalar (marka, model, yil, tip) VALUES ('Audi', 'e-tron', 2023, 'Elektrikli')")  
cursor.execute("INSERT INTO arabalar (marka, model, yil, tip) VALUES ('Mercedes-Benz', 'EQC', 2023, 'Elektrikli')")  
cursor.execute("INSERT INTO arabalar (marka, model, yil, tip) VALUES ('Jaguar', 'I-PACE', 2023, 'Elektrikli')")  
  
# Benzinli Arabalar  
cursor.execute("INSERT INTO arabalar (marka, model, yil, tip) VALUES ('Mercedes-Benz', 'S-Class', 2023, 'Benzinli')")  
cursor.execute("INSERT INTO arabalar (marka, model, yil, tip) VALUES ('BMW', '7 Series', 2023, 'Benzinli')")  
cursor.execute("INSERT INTO arabalar (marka, model, yil, tip) VALUES ('Lexus', 'LS', 2023, 'Benzinli')")  
cursor.execute("INSERT INTO arabalar (marka, model, yil, tip) VALUES ('Audi', 'A8', 2023, 'Benzinli')")  
cursor.execute("INSERT INTO arabalar (marka, model, yil, tip) VALUES ('Porsche', 'Panamera', 2023, 'Benzinli')")  
  
baglanti.commit()  
# Veritabanı bağlantısını kapat  
baglanti.close()

15.19 17.12.2023 güzel ama düzenlenmeli

import tkinter as tk  
from tkinter import ttk, simpledialog, messagebox  
import sqlite3  
import datetime  
from abc import ABC, abstractmethod  
class Araba(ABC):  
 def \_\_init\_\_(self, marka, model, yil):  
 self.marka = marka  
 self.model = model  
 self.\_yil = yil # Protected özellik  
 self.\_\_kilometre = 0 # Private özellik  
  
 def get\_yil(self):  
 return self.\_yil  
  
 def set\_kilometre(self, km):  
 if km >= 0:  
 self.\_\_kilometre = km  
 else:  
 print("Geçersiz kilometre değeri.")  
  
 def get\_kilometre(self):  
 return self.\_\_kilometre  
  
 @abstractmethod  
 def bakim(self):  
 pass  
  
class ElektrikliAraba(Araba):  
 def \_\_init\_\_(self, marka, model, yil, batarya\_durumu):  
 super().\_\_init\_\_(marka, model, yil)  
 self.batarya\_durumu = batarya\_durumu  
  
 def bakim(self):  
 return f"{self.marka} {self.model} elektrikli araba bakımı: Pil sağlığı kontrol ediliyor. Batarya Durumu: {self.batarya\_durumu}"  
  
class BenzinliAraba(Araba):  
 def \_\_init\_\_(self, marka, model, yil, yakit\_durumu):  
 super().\_\_init\_\_(marka, model, yil)  
 self.yakit\_durumu = yakit\_durumu  
  
 def bakim(self):  
 return f"{self.marka} {self.model} benzinli araba bakımı: Yağ değiştiriliyor. Yakıt Durumu: {self.yakit\_durumu}"  
class Kiralama:  
 def \_\_init\_\_(self, musteri, araba, kiralama\_tipi, sure):  
 self.musteri = musteri  
 self.araba = araba  
 self.kiralama\_tipi = kiralama\_tipi  
 self.sure = sure  
 self.baslangic\_tarihi = datetime.datetime.now()  
  
 def hesapla\_ucret(self):  
 if self.kiralama\_tipi == "günlük":  
 return self.sure \* 100  
 elif self.kiralama\_tipi == "saatlik":  
 return self.sure \* 20  
 else:  
 print("Geçersiz kiralama tipi")  
  
 # return ucret  
  
 def kiralama\_bitir(self):  
 bitis\_tarihi = datetime.datetime.now()  
 kiralama\_suresi = (bitis\_tarihi - self.baslangic\_tarihi).total\_seconds() / 3600  
  
 self.araba.set\_kilometre(self.araba.get\_kilometre() + kiralama\_suresi \* 10) # 10 km/saat varsayalım  
  
class AracKiralamaUygulamasi:  
 def \_\_init\_\_(self, root):  
 self.root = root  
 self.root.title("Araç Kiralama Uygulaması")  
  
 self.create\_database\_connection()  
  
 self.create\_main\_frame()  
  
 def create\_database\_connection(self):  
 self.baglanti = sqlite3.connect("rent-a.db")  
 self.cursor = self.baglanti.cursor()  
  
 def create\_main\_frame(self):  
 self.main\_frame = ttk.Frame(self.root, padding="20")  
 self.main\_frame.grid(row=0, column=0, sticky=(tk.W, tk.E, tk.N, tk.S))  
  
 self.label\_ad\_soyad = ttk.Label(self.main\_frame, text="Ad Soyad:")  
 self.label\_ad\_soyad.grid(row=0, column=0, sticky=tk.E)  
  
 self.entry\_ad\_soyad = ttk.Entry(self.main\_frame)  
 self.entry\_ad\_soyad.grid(row=0, column=1, sticky=(tk.W, tk.E))  
  
 self.button\_giris = ttk.Button(self.main\_frame, text="Giriş Yap", command=self.musteri\_giris)  
 self.button\_giris.grid(row=0, column=2, sticky=(tk.W, tk.E))  
  
 def musteri\_giris(self):  
 ad\_soyad = self.entry\_ad\_soyad.get()  
  
 if ad\_soyad:  
 musteri\_id = self.add\_musteri(ad\_soyad)  
 if musteri\_id:  
 self.musteri\_id = musteri\_id  
 self.label\_ad\_soyad.config(text=f"Merhaba, {ad\_soyad}")  
 self.button\_giris.config(state="disabled")  
 self.entry\_ad\_soyad.config(state="disabled")  
  
 self.show\_arac\_list()  
  
 def add\_musteri(self, ad\_soyad):  
 self.cursor.execute("INSERT INTO musteri (ad\_soyad) VALUES (?)", (ad\_soyad,))  
 self.baglanti.commit()  
 return self.cursor.lastrowid  
  
 def show\_arac\_list(self):  
 araclar = self.get\_arac\_list()  
  
 self.tree\_araclar = ttk.Treeview(self.main\_frame, columns=("Marka", "Model", "Yıl", "Tip", "Durum"), show="headings")  
 self.tree\_araclar.heading("Marka", text="Marka")  
 self.tree\_araclar.heading("Model", text="Model")  
 self.tree\_araclar.heading("Yıl", text="Yıl")  
 self.tree\_araclar.heading("Tip", text="Tip")  
 self.tree\_araclar.heading("Durum", text="Durum")  
  
 for arac in araclar:  
 self.tree\_araclar.insert("", "end", values=arac)  
  
 self.tree\_araclar.grid(row=1, column=0, columnspan=3, sticky=(tk.W, tk.E))  
  
 self.button\_kirala = ttk.Button(self.main\_frame, text="Araç Kirala", command=self.arac\_kirala)  
 self.button\_kirala.grid(row=2, column=0, columnspan=3, pady=10, sticky=(tk.W, tk.E))  
  
 def get\_arac\_list(self):  
 self.cursor.execute("SELECT arabalar.id, marka, model, yil, tip, durumu FROM arabalar WHERE durumu = 'müsait'")  
 return self.cursor.fetchall()  
  
 def arac\_kirala(self):  
 selected\_item = self.tree\_araclar.selection()  
 if selected\_item:  
 arac\_id = self.tree\_araclar.item(selected\_item, "values")[0]  
  
 kiralama\_tipi = self.show\_kiralama\_tipi\_dialog()  
 if kiralama\_tipi:  
 sure = self.show\_kiralama\_sure\_dialog()  
  
 if sure:  
 arac = self.get\_arac\_by\_id(arac\_id)  
  
 # Aracın bakım durumu GUI'de gösteriliyor  
 self.show\_arac\_bakim\_durumu(arac)  
  
 kiralama = Kiralama(self.musteri\_id, arac, kiralama\_tipi, sure)  
 ucret = kiralama.hesapla\_ucret()  
 if ucret != -1:  
 messagebox.showinfo("Başarılı", f"Araç kiralandı! Ücret: {ucret} TL")  
 kiralama.kiralama\_bitir()  
  
 def show\_arac\_bakim\_durumu(self, arac):  
 bakim\_durumu = arac.bakim()  
 messagebox.showinfo("Bakım Durumu", bakim\_durumu)  
  
 def show\_kiralama\_tipi\_dialog(self):  
 return simpledialog.askstring("Kiralama Tipi", "Kiralama Tipini Seçiniz (günlük/saatlik)")  
  
 def show\_kiralama\_sure\_dialog(self):  
 return simpledialog.askinteger("Kiralama Süresi", "Kiralama Süresini Giriniz (gün/saat)")  
  
 def get\_arac\_by\_id(self, arac\_id):  
 self.cursor.execute("SELECT \* FROM arabalar WHERE id = ?", (arac\_id,))  
 arac\_data = self.cursor.fetchone()  
  
 if arac\_data:  
 arac\_id, marka, model, yil, tip, durumu = arac\_data  
 if tip == 'Elektrikli':  
 return ElektrikliAraba(marka, model, yil, batarya\_durumu="Yüksek") # Batarya durumu varsayılan olarak "Yüksek"  
 elif tip == 'Benzinli':  
 return BenzinliAraba(marka, model, yil, yakit\_durumu="Doluluk Oranı: %100") # Yakıt durumu varsayılan olarak "Doluluk Oranı: %100"  
 else:  
 return None  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 root = tk.Tk()  
 app = AracKiralamaUygulamasi(root)  
 root.mainloop()

import sqlite3  
  
baglanti = sqlite3.connect("rent-a.db")  
cursor = baglanti.cursor()  
  
cursor.execute('''  
 CREATE TABLE IF NOT EXISTS musteri (  
 id INTEGER PRIMARY KEY AUTOINCREMENT,  
 ad\_soyad TEXT  
 )  
''')  
  
# Araba tablosunu oluştur  
cursor.execute('''  
 CREATE TABLE IF NOT EXISTS arabalar (  
 id INTEGER PRIMARY KEY AUTOINCREMENT,  
 marka TEXT,  
 model TEXT,  
 yil INTEGER,  
 tip TEXT,  
 durumu TEXT DEFAULT 'müsait'  
 )  
''')  
  
# Kiralama tablosunu oluştur  
cursor.execute('''  
 CREATE TABLE IF NOT EXISTS kiralama (  
 musteri\_id TEXT,  
 araba\_id INTEGER,  
 kiralama\_tipi TEXT,  
 sure INTEGER,  
 ucret INTEGER,  
 durumu TEXT ,  
 FOREIGN KEY (araba\_id) REFERENCES arabalar (id),  
 FOREIGN KEY (musteri\_id) REFERENCES musteri(id)  
   
 )  
''')  
  
  
  
# Elektrikli Arabalar  
cursor.execute("INSERT INTO arabalar (marka, model, yil, tip) VALUES ('Tesla', 'Model S', 2023, 'Elektrikli')")  
cursor.execute("INSERT INTO arabalar (marka, model, yil, tip) VALUES ('Porsche', 'Taycan', 2023, 'Elektrikli')")  
cursor.execute("INSERT INTO arabalar (marka, model, yil, tip) VALUES ('Audi', 'e-tron', 2023, 'Elektrikli')")  
cursor.execute("INSERT INTO arabalar (marka, model, yil, tip) VALUES ('Mercedes-Benz', 'EQC', 2023, 'Elektrikli')")  
cursor.execute("INSERT INTO arabalar (marka, model, yil, tip) VALUES ('Jaguar', 'I-PACE', 2023, 'Elektrikli')")  
  
# Benzinli Arabalar  
cursor.execute("INSERT INTO arabalar (marka, model, yil, tip) VALUES ('Mercedes-Benz', 'S-Class', 2023, 'Benzinli')")  
cursor.execute("INSERT INTO arabalar (marka, model, yil, tip) VALUES ('BMW', '7 Series', 2023, 'Benzinli')")  
cursor.execute("INSERT INTO arabalar (marka, model, yil, tip) VALUES ('Lexus', 'LS', 2023, 'Benzinli')")  
cursor.execute("INSERT INTO arabalar (marka, model, yil, tip) VALUES ('Audi', 'A8', 2023, 'Benzinli')")  
cursor.execute("INSERT INTO arabalar (marka, model, yil, tip) VALUES ('Porsche', 'Panamera', 2023, 'Benzinli')")  
  
baglanti.commit()  
# Veritabanı bağlantısını kapat  
baglanti.close()

Bakımı göstermeyi unutma yoksa bir işe yaramıyor

def show\_arac\_bakim\_durumu(self, arac):

bakim\_durumu = arac.bakim()

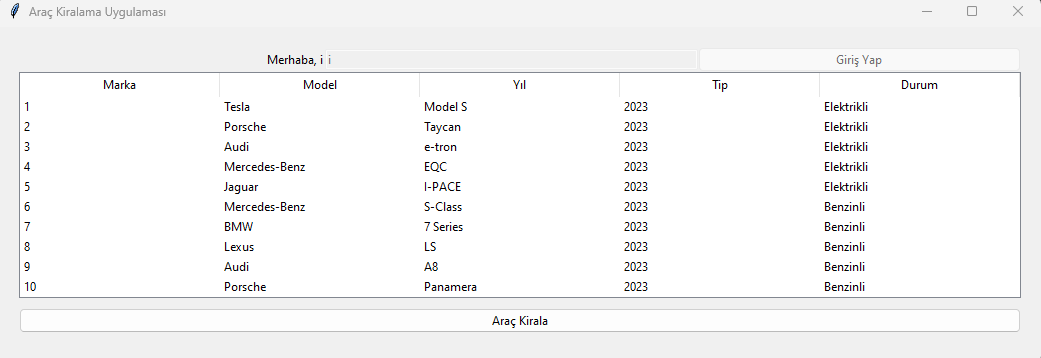
messagebox.showinfo("Bakım Durumu", bakim\_durumu)

23.26

import tkinter as tk  
from tkinter import ttk, simpledialog, messagebox  
import sqlite3  
import datetime  
from abc import ABC, abstractmethod  
class Araba(ABC):  
 def \_\_init\_\_(self, marka, model, yil):  
 self.marka = marka  
 self.model = model  
 self.\_yil = yil # Protected özellik  
 self.\_\_kilometre = 0 # Private özellik  
  
 def get\_yil(self):  
 return self.\_yil  
  
 def set\_kilometre(self, km):  
 if km >= 0:  
 self.\_\_kilometre = km  
 else:  
 print("Geçersiz kilometre değeri.")  
  
 def get\_kilometre(self):  
 return self.\_\_kilometre  
  
 @abstractmethod  
 def bakim(self):  
 pass  
  
class ElektrikliAraba(Araba):  
 def \_\_init\_\_(self, marka, model, yil, batarya\_durumu):  
 super().\_\_init\_\_(marka, model, yil)  
 self.batarya\_durumu = batarya\_durumu  
  
 def bakim(self):  
 return f"{self.marka} {self.model} elektrikli araba bakımı: Pil sağlığı kontrol ediliyor. Batarya Durumu: {self.batarya\_durumu}"  
  
class BenzinliAraba(Araba):  
 def \_\_init\_\_(self, marka, model, yil, yakit\_durumu):  
 super().\_\_init\_\_(marka, model, yil)  
 self.yakit\_durumu = yakit\_durumu  
  
 def bakim(self):  
 return f"{self.marka} {self.model} benzinli araba bakımı: Yağ değiştiriliyor. Yakıt Durumu: {self.yakit\_durumu}"  
class Kiralama:  
 def \_\_init\_\_(self, musteri, araba, kiralama\_tipi, sure):  
 self.musteri = musteri  
 self.araba = araba  
 self.kiralama\_tipi = kiralama\_tipi  
 self.sure = sure  
 self.baslangic\_tarihi = datetime.datetime.now()  
  
 def hesapla\_ucret(self):  
 if self.kiralama\_tipi == "günlük":  
 return self.sure \* 100  
 elif self.kiralama\_tipi == "saatlik":  
 return self.sure \* 20  
 else:  
 print("Geçersiz kiralama tipi")  
  
 # return ucret  
  
 def kiralama\_bitir(self):  
 bitis\_tarihi = datetime.datetime.now()  
 kiralama\_suresi = (bitis\_tarihi - self.baslangic\_tarihi).total\_seconds() / 3600  
  
 self.araba.set\_kilometre(self.araba.get\_kilometre() + kiralama\_suresi \* 10) # 10 km/saat varsayalım  
  
class AracKiralamaUygulamasi:  
 def \_\_init\_\_(self, root):  
 self.root = root  
 self.root.title("Araç Kiralama Uygulaması")  
  
 self.create\_database\_connection()  
  
 self.create\_main\_frame()  
  
 def create\_database\_connection(self):  
 self.baglanti = sqlite3.connect("rent-a.db")  
 self.cursor = self.baglanti.cursor()  
  
 def create\_main\_frame(self):  
 self.main\_frame = ttk.Frame(self.root, padding="20")  
 self.main\_frame.grid(row=0, column=0, sticky=(tk.W, tk.E, tk.N, tk.S))  
  
 self.label\_ad\_soyad = ttk.Label(self.main\_frame, text="Ad Soyad:")  
 self.label\_ad\_soyad.grid(row=0, column=0, sticky=tk.E)  
  
 self.entry\_ad\_soyad = ttk.Entry(self.main\_frame)  
 self.entry\_ad\_soyad.grid(row=0, column=1, sticky=(tk.W, tk.E))  
  
 self.button\_giris = ttk.Button(self.main\_frame, text="Giriş Yap", command=self.musteri\_giris)  
 self.button\_giris.grid(row=0, column=2, sticky=(tk.W, tk.E))  
  
 def musteri\_giris(self):  
 ad\_soyad = self.entry\_ad\_soyad.get()  
  
 if ad\_soyad:  
 musteri\_id = self.add\_musteri(ad\_soyad)  
 if musteri\_id:  
 self.musteri\_id = musteri\_id  
 self.label\_ad\_soyad.config(text=f"Merhaba, {ad\_soyad}")  
 self.button\_giris.config(state="disabled")  
 self.entry\_ad\_soyad.config(state="disabled")  
  
 self.show\_arac\_list()  
  
 def add\_musteri(self, ad\_soyad):  
 self.cursor.execute("INSERT INTO musteri (ad\_soyad) VALUES (?)", (ad\_soyad,))  
 self.baglanti.commit()  
 return self.cursor.lastrowid  
  
 def show\_arac\_list(self):  
 araclar = self.get\_arac\_list()  
  
 self.tree\_araclar = ttk.Treeview(self.main\_frame, columns=("Marka", "Model", "Yıl", "Tip", "Durum"), show="headings")  
 self.tree\_araclar.heading("Marka", text="Marka")  
 self.tree\_araclar.heading("Model", text="Model")  
 self.tree\_araclar.heading("Yıl", text="Yıl")  
 self.tree\_araclar.heading("Tip", text="Tip")  
 self.tree\_araclar.heading("Durum", text="Durum")  
  
 for arac in araclar:  
 self.tree\_araclar.insert("", "end", values=arac)  
  
 self.tree\_araclar.grid(row=1, column=0, columnspan=3, sticky=(tk.W, tk.E))  
  
 self.button\_kirala = ttk.Button(self.main\_frame, text="Araç Kirala", command=self.arac\_kirala)  
 self.button\_kirala.grid(row=2, column=0, columnspan=3, pady=10, sticky=(tk.W, tk.E))  
  
 def get\_arac\_list(self):  
 self.cursor.execute("SELECT arabalar.id, marka, model, yil, tip, durumu FROM arabalar WHERE durumu = 'müsait'")  
 return self.cursor.fetchall()  
  
 def arac\_kirala(self):  
 selected\_item = self.tree\_araclar.selection()  
 if selected\_item:  
 arac\_id = self.tree\_araclar.item(selected\_item, "values")[0]  
  
 kiralama\_tipi = self.show\_kiralama\_tipi\_dialog()  
 if kiralama\_tipi:  
 sure = self.show\_kiralama\_sure\_dialog()  
  
 if sure:  
 arac = self.get\_arac\_by\_id(arac\_id)  
  
 # Aracın bakım durumu GUI'de gösteriliyor  
 self.show\_arac\_bakim\_durumu(arac)  
  
 kiralama = Kiralama(self.musteri\_id, arac, kiralama\_tipi, sure)  
 ucret = kiralama.hesapla\_ucret()  
 if ucret != -1:  
 messagebox.showinfo("Başarılı", f"Araç kiralandı! Ücret: {ucret} TL")  
 kiralama.kiralama\_bitir()  
  
 def show\_arac\_bakim\_durumu(self, arac):  
 bakim\_durumu = arac.bakim()  
 messagebox.showinfo("Bakım Durumu", bakim\_durumu)  
  
 def show\_kiralama\_tipi\_dialog(self):  
 return simpledialog.askstring("Kiralama Tipi", "Kiralama Tipini Seçiniz (günlük/saatlik)")  
  
 def show\_kiralama\_sure\_dialog(self):  
 return simpledialog.askinteger("Kiralama Süresi", "Kiralama Süresini Giriniz (gün/saat)")  
  
 def get\_arac\_by\_id(self, arac\_id):  
 self.cursor.execute("SELECT \* FROM arabalar WHERE id = ?", (arac\_id,))  
 arac\_data = self.cursor.fetchone()  
  
 if arac\_data:  
 arac\_id, marka, model, yil, tip, durumu = arac\_data  
 if tip == 'Elektrikli':  
 return ElektrikliAraba(marka, model, yil, batarya\_durumu="Yüksek") # Batarya durumu varsayılan olarak "Yüksek"  
 elif tip == 'Benzinli':  
 return BenzinliAraba(marka, model, yil, yakit\_durumu="Doluluk Oranı: %100") # Yakıt durumu varsayılan olarak "Doluluk Oranı: %100"  
 else:  
 return None  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 root = tk.Tk()  
 app = AracKiralamaUygulamasi(root)  
 root.mainloop()

import sqlite3  
  
baglanti = sqlite3.connect("rent-a.db")  
cursor = baglanti.cursor()  
  
cursor.execute('''  
 CREATE TABLE IF NOT EXISTS musteri (  
 id INTEGER PRIMARY KEY AUTOINCREMENT,  
 ad\_soyad TEXT  
 )  
''')  
  
# Araba tablosunu oluştur  
cursor.execute('''  
 CREATE TABLE IF NOT EXISTS arabalar (  
 id INTEGER PRIMARY KEY AUTOINCREMENT,  
 marka TEXT,  
 model TEXT,  
 yil INTEGER,  
 tip TEXT,  
 durumu TEXT DEFAULT 'müsait'  
 )  
''')  
  
# Kiralama tablosunu oluştur  
cursor.execute('''  
 CREATE TABLE IF NOT EXISTS kiralama (  
 musteri\_id TEXT,  
 araba\_id INTEGER,  
 kiralama\_tipi TEXT,  
 sure INTEGER,  
 ucret INTEGER,  
 durumu TEXT ,  
 FOREIGN KEY (araba\_id) REFERENCES arabalar (id),  
 FOREIGN KEY (musteri\_id) REFERENCES musteri(id)  
   
 )  
''')  
  
  
  
# Elektrikli Arabalar  
cursor.execute("INSERT INTO arabalar (marka, model, yil, tip) VALUES ('Tesla', 'Model S', 2023, 'Elektrikli')")  
cursor.execute("INSERT INTO arabalar (marka, model, yil, tip) VALUES ('Porsche', 'Taycan', 2023, 'Elektrikli')")  
cursor.execute("INSERT INTO arabalar (marka, model, yil, tip) VALUES ('Audi', 'e-tron', 2023, 'Elektrikli')")  
cursor.execute("INSERT INTO arabalar (marka, model, yil, tip) VALUES ('Mercedes-Benz', 'EQC', 2023, 'Elektrikli')")  
cursor.execute("INSERT INTO arabalar (marka, model, yil, tip) VALUES ('Jaguar', 'I-PACE', 2023, 'Elektrikli')")  
  
# Benzinli Arabalar  
cursor.execute("INSERT INTO arabalar (marka, model, yil, tip) VALUES ('Mercedes-Benz', 'S-Class', 2023, 'Benzinli')")  
cursor.execute("INSERT INTO arabalar (marka, model, yil, tip) VALUES ('BMW', '7 Series', 2023, 'Benzinli')")  
cursor.execute("INSERT INTO arabalar (marka, model, yil, tip) VALUES ('Lexus', 'LS', 2023, 'Benzinli')")  
cursor.execute("INSERT INTO arabalar (marka, model, yil, tip) VALUES ('Audi', 'A8', 2023, 'Benzinli')")  
cursor.execute("INSERT INTO arabalar (marka, model, yil, tip) VALUES ('Porsche', 'Panamera', 2023, 'Benzinli')")  
  
baglanti.commit()  
# Veritabanı bağlantısını kapat  
baglanti.close()

metin, ekran görüntüsü, yazı tipi, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin, ekran görüntüsü, yazılım, bilgisayar simgesi içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

Gui düzenlenicek

Saatlik ve günlük yazısını kabul edilmeyen bir şeyse hata mesajı çıksın--------------

Giriş ekranı düzenlensin ----------------------------------------------------------------------

Oop doğru kullanılıyor mu

İlişki var mı kontrol et

Tablolara bak

Müsaitlik durumu falan güncellensin kiralanınca yeni kayıt eklensin

Araç müsait değilse gösterilmesin

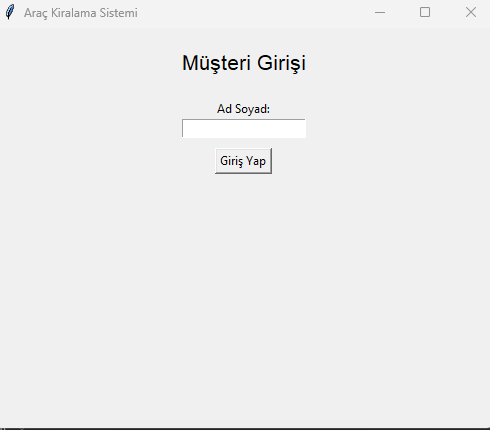
Araç iade et metodu da olsun ya da olmayadabilir

23.03 guiye karar vermeye çalışıyorum

from abc import ABC, abstractmethod  
import sqlite3  
import datetime  
  
class Araba(ABC):  
 def \_\_init\_\_(self, marka, model, yil):  
 self.marka = marka  
 self.model = model  
 self.\_yil = yil # Protected özellik  
 self.\_\_kilometre = 0 # Private özellik  
  
 def get\_yil(self):  
 return self.\_yil  
  
 def set\_kilometre(self, km):  
 if km >= 0:  
 self.\_\_kilometre = km  
 else:  
 print("Geçersiz kilometre değeri.")  
  
 def get\_kilometre(self):  
 return self.\_\_kilometre  
  
 @abstractmethod  
 def bakim(self):  
 pass  
  
class ElektrikliAraba(Araba):  
 def \_\_init\_\_(self, marka, model, yil, batarya\_durumu):  
 super().\_\_init\_\_(marka, model, yil)  
 self.batarya\_durumu = batarya\_durumu  
 def bakim(self):  
 return f"{self.marka} {self.model} elektrikli araba bakımı: Pil sağlığı kontrol ediliyor. Batarya Durumu: {self.batarya\_durumu}"  
  
class BenzinliAraba(Araba):  
 def \_\_init\_\_(self, marka, model, yil, yakit\_durumu):  
 super().\_\_init\_\_(marka, model, yil)  
 self.yakit\_durumu = yakit\_durumu  
 def bakim(self):  
 return f"{self.marka} {self.model} benzinli araba bakımı: Yağ değiştiriliyor. Yakıt Durumu: {self.yakit\_durumu}"  
  
class Kiralama:  
 def \_\_init\_\_(self, musteri, araba, kiralama\_tipi, sure):  
 self.musteri = musteri  
 self.araba = araba  
 self.kiralama\_tipi = kiralama\_tipi  
 self.sure = sure  
 self.baslangic\_tarihi = datetime.datetime.now()  
  
 def hesapla\_ucret(self):  
 if self.kiralama\_tipi == "günlük":  
 return self.sure \* 100  
 elif self.kiralama\_tipi == "saatlik":  
 return self.sure \* 20  
 else:  
 print("Geçersiz kiralama tipi")  
  
 # return ucret  
  
 def kiralama\_bitir(self):  
 bitis\_tarihi = datetime.datetime.now()  
 kiralama\_suresi = (bitis\_tarihi - self.baslangic\_tarihi).total\_seconds() / 3600  
  
 self.araba.set\_kilometre(self.araba.get\_kilometre() + kiralama\_suresi \* 10) # 10 km/saat varsayalım  
  
  
  
  
  
"""  
 kiralama = Kiralama("Ahmet", elektrikli\_araba, "günlük", 3)  
 ucret = kiralama.hesapla\_ucret()  
 if ucret != -1:  
 print(f"Ücret: {ucret} TL")"""

import sqlite3  
  
baglanti = sqlite3.connect("rent-a.db")  
cursor = baglanti.cursor()  
  
cursor.execute('''  
 CREATE TABLE IF NOT EXISTS musteri (  
 id INTEGER PRIMARY KEY AUTOINCREMENT,  
 ad\_soyad TEXT  
 )  
''')  
  
# Araba tablosunu oluştur  
cursor.execute('''  
 CREATE TABLE IF NOT EXISTS arabalar (  
 id INTEGER PRIMARY KEY AUTOINCREMENT,  
 marka TEXT,  
 model TEXT,  
 yil INTEGER,  
 tip TEXT,  
 durumu TEXT DEFAULT 'müsait'  
 )  
''')  
  
# Kiralama tablosunu oluştur  
cursor.execute('''  
 CREATE TABLE IF NOT EXISTS kiralama (  
 musteri\_id TEXT,  
 araba\_id INTEGER,  
 kiralama\_tipi TEXT,  
 sure INTEGER,  
 ucret INTEGER,  
 durumu TEXT ,  
 FOREIGN KEY (araba\_id) REFERENCES arabalar (id),  
 FOREIGN KEY (musteri\_id) REFERENCES musteri(id)  
   
 )  
''')  
  
  
  
# Elektrikli Arabalar  
cursor.execute("INSERT INTO arabalar (marka, model, yil, tip) VALUES ('Tesla', 'Model S', 2023, 'Elektrikli')")  
cursor.execute("INSERT INTO arabalar (marka, model, yil, tip) VALUES ('Porsche', 'Taycan', 2023, 'Elektrikli')")  
cursor.execute("INSERT INTO arabalar (marka, model, yil, tip) VALUES ('Audi', 'e-tron', 2023, 'Elektrikli')")  
cursor.execute("INSERT INTO arabalar (marka, model, yil, tip) VALUES ('Mercedes-Benz', 'EQC', 2023, 'Elektrikli')")  
cursor.execute("INSERT INTO arabalar (marka, model, yil, tip) VALUES ('Jaguar', 'I-PACE', 2023, 'Elektrikli')")  
  
# Benzinli Arabalar  
cursor.execute("INSERT INTO arabalar (marka, model, yil, tip) VALUES ('Mercedes-Benz', 'S-Class', 2023, 'Benzinli')")  
cursor.execute("INSERT INTO arabalar (marka, model, yil, tip) VALUES ('BMW', '7 Series', 2023, 'Benzinli')")  
cursor.execute("INSERT INTO arabalar (marka, model, yil, tip) VALUES ('Lexus', 'LS', 2023, 'Benzinli')")  
cursor.execute("INSERT INTO arabalar (marka, model, yil, tip) VALUES ('Audi', 'A8', 2023, 'Benzinli')")  
cursor.execute("INSERT INTO arabalar (marka, model, yil, tip) VALUES ('Porsche', 'Panamera', 2023, 'Benzinli')")  
  
baglanti.commit()  
# Veritabanı bağlantısını kapat  
baglanti.close()

import tkinter as tk  
from tkinter import messagebox  
from main import BenzinliAraba, ElektrikliAraba, Kiralama  
  
  
class ArabaKiralamaGUI:  
 def \_\_init\_\_(self, root):  
 self.root = root  
 self.root.title("Araç Kiralama Sistemi")  
 self.root.geometry("500x400")  
  
 # Başlangıç ekranı  
 self.baslangic\_ekrani()  
  
 def baslangic\_ekrani(self):  
 self.clear\_window()  
  
 label = tk.Label(self.root, text="Müşteri Girişi", font=("Helvetica", 16))  
 label.pack(pady=20)  
  
 self.ad\_soyad\_label = tk.Label(self.root, text="Ad Soyad:")  
 self.ad\_soyad\_label.pack()  
  
 self.ad\_soyad\_entry = tk.Entry(self.root)  
 self.ad\_soyad\_entry.pack()  
  
 giris\_button = tk.Button(self.root, text="Giriş Yap", command=self.musteri\_girisi)  
 giris\_button.pack(pady=10)  
  
 def musteri\_girisi(self):  
 ad\_soyad = self.ad\_soyad\_entry.get()  
  
 if ad\_soyad:  
 self.clear\_window()  
 self.kiralama\_ekrani(ad\_soyad)  
 else:  
 messagebox.showerror("Hata", "Lütfen ad soyadınızı girin.")  
  
 def kiralama\_ekrani(self, ad\_soyad):  
 label = tk.Label(self.root, text=f"Hoşgeldiniz, {ad\_soyad}!", font=("Helvetica", 16))  
 label.pack(pady=20)  
  
 label\_araba = tk.Label(self.root, text="Araba Seçiniz:")  
 label\_araba.pack()  
  
 araba\_var = tk.StringVar(self.root)  
 araba\_var.set("Elektrikli") # Başlangıçta Elektrikli Araba seçili  
  
 araba\_secim = tk.OptionMenu(self.root, araba\_var, "Elektrikli", "Benzinli")  
 araba\_secim.pack(pady=10)  
  
 label\_tip = tk.Label(self.root, text="Kiralama Tipi:")  
 label\_tip.pack()  
  
 tip\_var = tk.StringVar(self.root)  
 tip\_var.set("günlük") # Başlangıçta Günlük seçili  
  
 tip\_secim = tk.OptionMenu(self.root, tip\_var, "günlük", "saatlik")  
 tip\_secim.pack(pady=10)  
  
 label\_sure = tk.Label(self.root, text="Kiralama Süresi:")  
 label\_sure.pack()  
  
 sure\_entry = tk.Entry(self.root)  
 sure\_entry.pack(pady=10)  
  
 kiralama\_button = tk.Button(self.root, text="Kiralama Yap", command=lambda: self.kiralama\_yap(ad\_soyad, araba\_var.get(), tip\_var.get(), sure\_entry.get()))  
 kiralama\_button.pack(pady=20)  
  
 def kiralama\_yap(self, musteri, araba\_tipi, kiralama\_tipi, sure):  
 try:  
 sure = int(sure)  
 except ValueError:  
 messagebox.showerror("Hata", "Geçersiz süre değeri.")  
 return  
  
 if araba\_tipi == "Elektrikli":  
 araba = ElektrikliAraba("Tesla", "Model S", 2023, 90)  
 elif araba\_tipi == "Benzinli":  
 araba = BenzinliAraba("Toyota", "Corolla", 2023, "Çalışabilir")  
 else:  
 messagebox.showerror("Hata", "Geçersiz araba tipi.")  
 return  
  
 kiralama = Kiralama(musteri, araba, kiralama\_tipi, sure)  
 ucret = kiralama.hesapla\_ucret()  
  
 if ucret != -1:  
 messagebox.showinfo("Başarılı", f"Kiralama başarılı! Ücret: {ucret} TL")  
  
 def clear\_window(self):  
 for widget in self.root.winfo\_children():  
 widget.destroy()  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 root = tk.Tk()  
 app = ArabaKiralamaGUI(root)  
 root.mainloop()

 metin, ekran görüntüsü, yazılım, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

import tkinter as tk  
from tkinter import ttk, messagebox  
from main import BenzinliAraba, ElektrikliAraba, Kiralama  
class AracKiralamaGUI:  
 def \_\_init\_\_(self, root):  
 self.root = root  
 self.root.title("Araç Kiralama Sistemi")  
  
 self.label = ttk.Label(root, text="Müşteri Adı Soyadı:")  
 self.label.grid(row=0, column=0, padx=10, pady=10)  
  
 self.musteri\_entry = ttk.Entry(root)  
 self.musteri\_entry.grid(row=0, column=1, padx=10, pady=10)  
  
 self.giris\_button = ttk.Button(root, text="Giriş Yap", command=self.musteri\_girisi)  
 self.giris\_button.grid(row=0, column=2, padx=10, pady=10)  
  
 def musteri\_girisi(self):  
 musteri\_adi\_soyadi = self.musteri\_entry.get()  
  
 if musteri\_adi\_soyadi:  
 self.root.destroy() # Giriş yapılırsa ana pencereyi kapat  
  
 kiralama\_penceresi = tk.Tk()  
 kiralama\_penceresi.title("Kiralama İşlemleri")  
  
 self.kiralama\_arayuzu(kiralama\_penceresi, musteri\_adi\_soyadi)  
 else:  
 messagebox.showwarning("Uyarı", "Lütfen adınızı ve soyadınızı giriniz.")  
  
 def kiralama\_arayuzu(self, root, musteri\_adi\_soyadi):  
 self.musteri\_adi\_soyadi = musteri\_adi\_soyadi  
  
 label = ttk.Label(root, text=f"{musteri\_adi\_soyadi}, kiralama işlemleri için lütfen aşağıdaki bilgileri giriniz:")  
 label.grid(row=0, column=0, columnspan=3, padx=10, pady=10)  
  
 araba\_label = ttk.Label(root, text="Araba Seçimi:")  
 araba\_label.grid(row=1, column=0, padx=10, pady=10)  
  
 araba\_combobox = ttk.Combobox(root, values=["Elektrikli Araba", "Benzinli Araba"])  
 araba\_combobox.grid(row=1, column=1, padx=10, pady=10)  
  
 tip\_label = ttk.Label(root, text="Kiralama Tipi:")  
 tip\_label.grid(row=2, column=0, padx=10, pady=10)  
  
 tip\_combobox = ttk.Combobox(root, values=["Günlük", "Saatlik"])  
 tip\_combobox.grid(row=2, column=1, padx=10, pady=10)  
  
 sure\_label = ttk.Label(root, text="Kiralama Süresi:")  
 sure\_label.grid(row=3, column=0, padx=10, pady=10)  
  
 sure\_entry = ttk.Entry(root)  
 sure\_entry.grid(row=3, column=1, padx=10, pady=10)  
  
 kiralama\_button = ttk.Button(root, text="Kiralama Yap", command=lambda: self.kiralama\_yap(root, araba\_combobox.get(), tip\_combobox.get(), sure\_entry.get()))  
 kiralama\_button.grid(row=4, column=0, columnspan=2, padx=10, pady=10)  
  
 def kiralama\_yap(self, root, araba\_tipi, kiralama\_tipi, sure):  
 try:  
 sure = int(sure)  
 except ValueError:  
 messagebox.showwarning("Uyarı", "Geçersiz süre girişi. Lütfen bir sayı girin.")  
 return  
  
 if araba\_tipi == "Elektrikli Araba":  
 araba = ElektrikliAraba("Tesla", "Model S", 2023)  
 elif araba\_tipi == "Benzinli Araba":  
 araba = BenzinliAraba("Toyota", "Corolla", 2023)  
 else:  
 messagebox.showwarning("Uyarı", "Geçersiz araba tipi seçimi.")  
 return  
  
 kiralama = Kiralama(self.musteri\_adi\_soyadi, araba, kiralama\_tipi, sure)  
 ucret = kiralama.hesapla\_ucret()  
  
 if ucret != -1:  
 messagebox.showinfo("Başarılı", f"Kiralama işlemi tamamlandı. Ücret: {ucret} TL")  
 root.destroy()  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 root = tk.Tk()  
 gui = AracKiralamaGUI(root)  
 root.mainloop()

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu hatalı

import tkinter as tk  
from tkinter import ttk  
import sqlite3  
import datetime  
  
class AracKiralamaUygulamasi:  
 def \_\_init\_\_(self, root):  
 self.root = root  
 self.root.title("Araç Kiralama Uygulaması")  
  
 self.create\_database\_connection()  
  
 self.create\_main\_frame()  
  
 def create\_database\_connection(self):  
 self.baglanti = sqlite3.connect("rent-a.db")  
 self.cursor = self.baglanti.cursor()  
  
 def create\_main\_frame(self):  
 self.main\_frame = ttk.Frame(self.root, padding="20")  
 self.main\_frame.grid(row=0, column=0, sticky=(tk.W, tk.E, tk.N, tk.S))  
  
 self.label\_ad\_soyad = ttk.Label(self.main\_frame, text="Ad Soyad:")  
 self.label\_ad\_soyad.grid(row=0, column=0, sticky=tk.E)  
  
 self.entry\_ad\_soyad = ttk.Entry(self.main\_frame)  
 self.entry\_ad\_soyad.grid(row=0, column=1, sticky=(tk.W, tk.E))  
  
 self.button\_giris = ttk.Button(self.main\_frame, text="Giriş Yap", command=self.musteri\_giris)  
 self.button\_giris.grid(row=0, column=2, sticky=(tk.W, tk.E))  
  
 self.create\_arac\_list()  
  
 def create\_arac\_list(self):  
 araclar = self.get\_arac\_list()  
  
 self.tree\_araclar = ttk.Treeview(self.main\_frame, columns=("Marka", "Model", "Yıl", "Tip"), show="headings")  
 self.tree\_araclar.heading("Marka", text="Marka")  
 self.tree\_araclar.heading("Model", text="Model")  
 self.tree\_araclar.heading("Yıl", text="Yıl")  
 self.tree\_araclar.heading("Tip", text="Tip")  
  
 for arac in araclar:  
 self.tree\_araclar.insert("", "end", values=arac)  
  
 self.tree\_araclar.grid(row=1, column=0, columnspan=3, sticky=(tk.W, tk.E))  
  
 self.button\_kirala = ttk.Button(self.main\_frame, text="Araç Kirala", command=self.arac\_kirala)  
 self.button\_kirala.grid(row=2, column=0, columnspan=3, pady=10, sticky=(tk.W, tk.E))  
  
 def musteri\_giris(self):  
 ad\_soyad = self.entry\_ad\_soyad.get()  
  
 if ad\_soyad:  
 musteri\_id = self.add\_musteri(ad\_soyad)  
 if musteri\_id:  
 self.musteri\_id = musteri\_id  
 self.label\_ad\_soyad.config(text=f"Merhaba, {ad\_soyad}")  
 self.button\_giris.config(state="disabled")  
 self.entry\_ad\_soyad.config(state="disabled")  
  
 def add\_musteri(self, ad\_soyad):  
 self.cursor.execute("INSERT INTO musteri (ad\_soyad) VALUES (?)", (ad\_soyad,))  
 self.baglanti.commit()  
 return self.cursor.lastrowid  
  
 def get\_arac\_list(self):  
 self.cursor.execute("SELECT \* FROM arabalar")  
 return self.cursor.fetchall()  
  
 def arac\_kirala(self):  
 selected\_item = self.tree\_araclar.selection()  
 if selected\_item:  
 arac\_id = self.tree\_araclar.item(selected\_item, "values")[0]  
 kiralama\_tipi = "günlük" # Örnek olarak "günlük" seçildi.  
 sure = 1 # Örnek olarak kiralama süresi 1 gün olarak seçildi.  
  
 self.kiralama\_islemi(self.musteri\_id, arac\_id, kiralama\_tipi, sure)  
 print("Araç kiralandı!")  
  
 def kiralama\_islemi(self, musteri\_id, arac\_id, kiralama\_tipi, sure):  
 self.cursor.execute("INSERT INTO kiralama (musteri\_id, arac\_id, kiralama\_tipi, sure, ucret) VALUES (?, ?, ?, ?, ?)",  
 (musteri\_id, arac\_id, kiralama\_tipi, sure, sure \* 100)) # Günlük ücret 100 TL olarak varsayıldı.  
 self.baglanti.commit()  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 root = tk.Tk()  
 app = AracKiralamaUygulamasi(root)  
 root.mainloop()

metin, ekran görüntüsü, yazılım, ekran, görüntüleme içeren bir resim

Açıklama otomatik olarak oluşturuldu

Hatalı

-------------------------------

import tkinter as tk  
from tkinter import ttk  
  
class ArabaKiralamaGUI:  
 def \_\_init\_\_(self, root):  
 self.root = root  
 self.root.title("Araç Kiralama Sistemi")  
  
 self.musteri\_id = None  
 self.musteri\_ad\_soyad = None  
  
 self.create\_musteri\_giris()  
  
 def create\_musteri\_giris(self):  
 self.musteri\_giris\_frame = ttk.Frame(self.root)  
 self.musteri\_giris\_frame.grid(row=0, column=0, padx=20, pady=20)  
  
 label = ttk.Label(self.musteri\_giris\_frame, text="Ad ve Soyad:")  
 label.grid(row=0, column=0, padx=10, pady=10)  
  
 self.ad\_soyad\_entry = ttk.Entry(self.musteri\_giris\_frame)  
 self.ad\_soyad\_entry.grid(row=0, column=1, padx=10, pady=10)  
  
 giris\_button = ttk.Button(self.musteri\_giris\_frame, text="Giriş Yap", command=self.giris\_yap)  
 giris\_button.grid(row=1, column=0, columnspan=2, pady=10)  
  
 def create\_kiralama\_arayuz(self):  
 self.musteri\_giris\_frame.destroy()  
  
 self.kiralama\_frame = ttk.Frame(self.root)  
 self.kiralama\_frame.grid(row=0, column=0, padx=20, pady=20)  
  
 label = ttk.Label(self.kiralama\_frame, text=f"Hoşgeldiniz, {self.musteri\_ad\_soyad}!")  
 label.grid(row=0, column=0, columnspan=2, pady=10)  
  
 label = ttk.Label(self.kiralama\_frame, text="Kiralama Tipi:")  
 label.grid(row=1, column=0, padx=10, pady=10)  
  
 self.kiralama\_tipi\_combobox = ttk.Combobox(self.kiralama\_frame, values=["Günlük", "Saatlik"])  
 self.kiralama\_tipi\_combobox.grid(row=1, column=1, padx=10, pady=10)  
  
 label = ttk.Label(self.kiralama\_frame, text="Kiralama Süresi:")  
 label.grid(row=2, column=0, padx=10, pady=10)  
  
 self.kiralama\_suresi\_entry = ttk.Entry(self.kiralama\_frame)  
 self.kiralama\_suresi\_entry.grid(row=2, column=1, padx=10, pady=10)  
  
 kiralama\_button = ttk.Button(self.kiralama\_frame, text="Kirala", command=self.kirala)  
 kiralama\_button.grid(row=3, column=0, columnspan=2, pady=10)  
  
 def giris\_yap(self):  
 ad\_soyad = self.ad\_soyad\_entry.get()  
 if ad\_soyad:  
 # Burada müşteri bilgilerini kontrol edebilir ve giriş işlemini gerçekleştirebilirsiniz.  
 # Örneğin, bir veritabanında müşteri bilgilerini kontrol edebilirsiniz.  
 # Şu an sadece ad ve soyad bilgisini alıyoruz.  
 self.musteri\_ad\_soyad = ad\_soyad  
 self.create\_kiralama\_arayuz()  
  
 def kirala(self):  
 kiralama\_tipi = self.kiralama\_tipi\_combobox.get()  
 kiralama\_suresi = self.kiralama\_suresi\_entry.get()  
  
 if kiralama\_tipi and kiralama\_suresi:  
 # Burada kiralama işlemlerini gerçekleştirebilirsiniz.  
 # Örneğin, Kiralama sınıfını kullanarak işlemleri gerçekleştirebilirsiniz.  
 pass  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 root = tk.Tk()  
 gui = ArabaKiralamaGUI(root)  
 root.mainloop()

 metin, ekran görüntüsü, yazı tipi, ekran, görüntüleme içeren bir resim

Açıklama otomatik olarak oluşturuldu

21.01 gui ve veritabanı yok

from abc import ABC, abstractmethod  
import sqlite3  
import datetime  
  
class Araba(ABC):  
 def \_\_init\_\_(self, marka, model, yil):  
 self.marka = marka  
 self.model = model  
 self.\_yil = yil # Protected özellik  
 self.\_\_kilometre = 0 # Private özellik  
  
 def get\_yil(self):  
 return self.\_yil  
  
 def set\_kilometre(self, km):  
 if km >= 0:  
 self.\_\_kilometre = km  
 else:  
 print("Geçersiz kilometre değeri.")  
  
 def get\_kilometre(self):  
 return self.\_\_kilometre  
  
 @abstractmethod  
 def bakim(self):  
 pass  
  
class ElektrikliAraba(Araba):  
 def \_\_init\_\_(self, marka, model, yil, batarya\_durumu):  
 super().\_\_init\_\_(marka, model, yil)  
 self.batarya\_durumu = batarya\_durumu  
 def bakim(self):  
 return f"{self.marka} {self.model} elektrikli araba bakımı: Pil sağlığı kontrol ediliyor. Batarya Durumu: {self.batarya\_durumu}"  
class BenzinliAraba(Araba):  
 def \_\_init\_\_(self, marka, model, yil, yakit\_durumu):  
 super().\_\_init\_\_(marka, model, yil)  
 self.yakit\_durumu = yakit\_durumu  
 def bakim(self):  
 return f"{self.marka} {self.model} benzinli araba bakımı: Yağ değiştiriliyor. Yakıt Durumu: {self.yakit\_durumu}"  
  
class Kiralama:  
 def \_\_init\_\_(self, musteri, araba, tip, sure):  
 self.musteri = musteri  
 self.araba = araba  
 self.tip = tip  
 self.sure = sure  
 self.baslangic\_tarihi = datetime.datetime.now()  
  
 def hesapla\_ucret(self):  
 if self.tip == "günlük":  
 return self.sure \* 100  
 elif self.tip == "saatlik":  
 return self.sure \* 20  
 else:  
 print("Geçersiz kiralama tipi")  
  
 return ucret  
  
 def kiralama\_bitir(self):  
 bitis\_tarihi = datetime.datetime.now()  
 kiralama\_suresi = (bitis\_tarihi - self.baslangic\_tarihi).total\_seconds() / 3600  
  
 self.araba.set\_kilometre(self.araba.get\_kilometre() + kiralama\_suresi \* 10) # 10 km/saat varsayalım  
  
  
class Veritabani:  
 def \_\_init\_\_(self):  
 self.baglanti = sqlite3.connect("araba\_kiralama.db")  
 self.cursor = self.baglanti.cursor()  
 self.cursor.execute('''  
 CREATE TABLE IF NOT EXISTS kiralama (  
 musteri TEXT,  
 araba\_marka TEXT,  
 araba\_model TEXT,  
 araba\_yil INTEGER,  
 tip TEXT,  
 sure INTEGER,  
 ucret INTEGER  
 )  
 ''')  
 self.baglanti.commit()  
  
 def kiralama\_kaydet(self, musteri, araba, tip, sure, ucret):  
 araba\_marka, araba\_model, araba\_yil = araba.marka, araba.model, araba.get\_yil()  
 self.cursor.execute("INSERT INTO kiralama VALUES (?, ?, ?, ?, ?, ?, ?)",  
 (musteri, araba\_marka, araba\_model, araba\_yil, tip, sure, ucret))  
 self.baglanti.commit()  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 elektrikli\_araba = ElektrikliAraba("Tesla", "Model S", 2022, 90)  
 benzinli\_araba = BenzinliAraba("Toyota", "Corolla", 2022, "Çalışabilir")  
  
 elektrikli\_araba.set\_kilometre(50000)  
 benzinli\_araba.set\_kilometre(60000)  
  
 print(elektrikli\_araba.bakim())  
 print(benzinli\_araba.bakim())  
  
 kiralama = Kiralama("Ahmet", elektrikli\_araba, "günlük", 2)  
 ucret = kiralama.hesapla\_ucret()  
  
 if ucret != -1:  
 print(f"Ücret: {ucret} TL")  
 vt = Veritabani()  
 vt.kiralama\_kaydet("Ahmet", elektrikli\_araba, "günlük", 2, ucret)  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 elektrikli\_araba = ElektrikliAraba("Tesla", "Model S", 2023)  
 print(elektrikli\_araba.bakim())  
  
 benzinli\_araba = BenzinliAraba("Toyota", "Corolla", 2023)  
 print(benzinli\_araba.bakim())  
  
 kiralama = Kiralama("Ahmet", elektrikli\_araba, "günlük", 3)  
 ucret = kiralama.hesapla\_ucret()  
 if ucret != -1:  
 print(f"Ücret: {ucret} TL")