Масленников Владислав

**Отчет  
 Тема:Парковка машин**

**Цель работы**: разработка программы для учета автомобилей, парковок

**Задачи**:

Описать классы "Автомобиль" и "Стоянка".

Класс "Автомобиль" должен содержать информацию о машине: Марку, Модель, Цвет, Номер, Время прибытия.

Добавьте возможность добавления машин на стоянку, просмотра информации о них и их вывоз.

Предусмотреть защиту от ввода некорректных данных для всех членов всех классов

Предусмотреть повторное предложение Пользователю на ввод данных, в случае их некорректности

Предусмотреть назначение Автомобилям уникальных идентификаторов

Предусмотреть выбор Предприятия на приема и переода работников

Написать класс для сохранения состояния программы в текстовом файле, на примере стоянки:

при изменении данных перезаписывать файл,

при новом запуске программы восстанавливать данные из файла

написать класс для логирования событий в текстовый файл  
  
using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

class Car

{

public string Brand { get; set; }

public string Model { get; set; }

public string Color { get; set; }

public string LicensePlate { get; set; }

public DateTime ArrivalTime { get; set; }

public Car(string brand, string model, string color, string licensePlate, DateTime arrivalTime)

{

Brand = brand;

Model = model;

Color = color;

LicensePlate = licensePlate;

ArrivalTime = arrivalTime;

}

}

class Parking

{

private List<Car> cars = new List<Car>();

private FileHandler fileHandler = new FileHandler();

private Logger logger = new Logger();

public Parking()

{

// При создании экземпляра Parking загружаем данные из файла

cars = fileHandler.ReadFromFile();

}

public void AddCar(Car car)

{

cars.Add(car);

fileHandler.SaveToFile(cars);

logger.LogEntry($"Прибытие автомобиля {car.Brand} {car.Model}");

}

public Car GetCarByLicensePlate(string licensePlate)

{

return cars.FirstOrDefault(car => car.LicensePlate == licensePlate);

}

public void RemoveCar(Car car)

{

cars.Remove(car);

fileHandler.SaveToFile(cars);

logger.LogEntry($"Убытие автомобиля {car.Brand} {car.Model}");

}

public void ViewCars()

{

foreach (var car in cars)

{

Console.WriteLine($"\nМарка: {car.Brand}\nМодель: {car.Model}\nЦвет: {car.Color}\nНомера: {car.LicensePlate}\nВремя прибытия: {car.ArrivalTime}\n");

}

}

public void RunMenu()

{

while (true)

{

Console.WriteLine("Выберите действие:");

Console.WriteLine("1. Добавить автомобиль на стоянку");

Console.WriteLine("2. Посмотреть информацию о машинах на стоянке");

Console.WriteLine("3. Вывезти машину с парковки");

Console.WriteLine("4. Выход");

string choice = Console.ReadLine();

switch (choice)

{

case "1":

Console.WriteLine("Введите марку автомобиля:");

string brand = Console.ReadLine();

Console.WriteLine("Введите модель автомобиля:");

string model = Console.ReadLine();

Console.WriteLine("Введите цвет автомобиля:");

string color = Console.ReadLine();

string licensePlate;

do

{

Console.WriteLine("Введите номер автомобиля:");

licensePlate = Console.ReadLine();

} while (string.IsNullOrWhiteSpace(licensePlate));

DateTime arrivalTime;

string arrivalTimeString;

do

{

Console.WriteLine("Введите время прибытия (гггг-мм-дд чч:мм):");

arrivalTimeString = Console.ReadLine();

} while (!DateTime.TryParse(arrivalTimeString, out arrivalTime));

Car newCar = new Car(brand, model, color, licensePlate, arrivalTime);

AddCar(newCar);

Console.WriteLine("Автомобиль успешно добавлен на стоянку.");

break;

case "2":

Console.WriteLine("Информация о машинах на стоянке:");

ViewCars();

break;

case "3":

Console.WriteLine("Введите номер машины, чтобы вывезти её с парковки:");

string carLicensePlateToExit = Console.ReadLine();

Car carToExit = GetCarByLicensePlate(carLicensePlateToExit);

if (carToExit != null)

{

RemoveCar(carToExit);

Console.WriteLine($"Автомобиль {carLicensePlateToExit} успешно вывезен с парковки.");

}

else

{

Console.WriteLine("Автомобиль не найден на стоянке.");

}

break;

case "4":

Environment.Exit(0);

break;

default:

Console.WriteLine("Неверный выбор. Пожалуйста, выберите снова.");

break;

}

}

}

}

class FileHandler

{

private string filePath = "parking\_data.txt";

public void SaveToFile(List<Car> cars)

{

using (StreamWriter writer = new StreamWriter(filePath))

{

foreach (var car in cars)

{

writer.WriteLine($"{car.Brand};{car.Model};{car.Color};{car.LicensePlate};{car.ArrivalTime}");

}

}

}

public List<Car> ReadFromFile()

{

List<Car> cars = new List<Car>();

if (File.Exists(filePath))

{

using (StreamReader reader = new StreamReader(filePath))

{

string line;

while ((line = reader.ReadLine()) != null)

{

string[] data = line.Split(';');

if (data.Length == 5)

{

Car car = new Car(data[0], data[1], data[2], data[3], DateTime.Parse(data[4]));

cars.Add(car);

}

}

}

}

return cars;

}

}

class Logger

{

private string logFilePath = "parking\_log.txt";

public void LogEntry(string logEntry)

{

using (StreamWriter writer = new StreamWriter(logFilePath, true))

{

writer.WriteLine($"{DateTime.Now}: {logEntry}");

}

}

}

class Program

{

static void Main(string[] args)

{

Parking parking = new Parking();

parking.RunMenu();

}

}