Ministerul Educației și Cercetării al Republicii Moldova
Colegiul Universității Tehnice a Moldovei
Administrarea Aplicațiilor Web

RAPORT

Lucrare de laborator nr.3

Disciplina: Asistență pentru programarea orientată pe obiecte

Tema: Sistem de gestionare a parcării

A efectuat: Tintiuc Cătălin

Chișinău 2025

```cpp
#include <iostream>
#include <fstream>
#include <vector>
#include <string>

using namespace std;

class Vehicul {
protected:
    string numarInmatriculare;
    int oraIntrare, oraIesire;
public:
    Vehicul(string numar, int intrare, int iesire) : numarInmatriculare(numar), oraIntrare(intrare), oraIesire(iesire) {}
    virtual ~Vehicul() {}
    int calculTimp() const { return oraIesire - oraIntrare; }
    virtual double calculTarif() const = 0;
    virtual void afisare() const = 0;
};

class Masina : public Vehicul {
public:
    Masina(string numar, int intrare, int iesire) : Vehicul(numar, intrare, iesire) {}
    double calculTarif() const override { return calculTimp() * 5; }
    void afisare() const override {
        cout << "Masina " << numarInmatriculare << " - " << calculTimp() << " ore - " << calculTarif() << " lei" << endl;
    }
};

class Motocicleta : public Vehicul {
public:
    Motocicleta(string numar, int intrare, int iesire) : Vehicul(numar, intrare, iesire) {}
    double calculTarif() const override { return calculTimp() * 3; }
    void afisare() const override {
        cout << "Motocicleta " << numarInmatriculare << " - " << calculTimp() << " ore - " << calculTarif() << " lei" << endl;
    }
};
```

```cpp
class Camion : public Vehicul {
public:
    Camion(string numar, int intrare, int iesire) : Vehicul(numar, intrare, iesire) {}
    double calculTarif() const override { return calculTimp() * 10; }
    void afisare() const override {
        cout << "Camion " << numarInmatriculare << " - " << calculTimp() << " ore - " << calculTarif() << " lei" << endl;
    }
};

class Bicicleta : public Vehicul {
public:
    Bicicleta(string numar, int intrare, int iesire) : Vehicul(numar, intrare, iesire) {}
    double calculTarif() const override {
        int timp = calculTimp();
        return (timp <= 2) ? 0 : (timp - 2) * 2;
    }
    void afisare() const override {
        cout << "Bicicleta " << numarInmatriculare << " - " << calculTimp() << " ore - " << calculTarif() << " lei" << endl;
    }
};

void citireFisier(vector<Vehicul*>& vehicule, const string& numeFisier) {
    ifstream fisier(numeFisier);
    if (!fisier) {
        cerr << "Eroare la deschiderea fisierului!" << endl;
        return;
    }
    string tip, numar;
    int intrare, iesire;
    while (fisier >> tip >> numar >> intrare >> iesire) {
        if (tip == "Masina") vehicule.push_back(new Masina(numar, intrare, iesire));
        else if (tip == "Motocicleta") vehicule.push_back(new Motocicleta(numar, intrare, iesire));
        else if (tip == "Camion") vehicule.push_back(new Camion(numar, intrare, iesire));
        else if (tip == "Bicicleta") vehicule.push_back(new Bicicleta(numar, intrare, iesire));
    }
    fisier.close();
}
```

```cpp
void afisareVehicule(const vector<Vehicul*>& vehicule) {
    for (const auto& vehicul : vehicule) {
        vehicul->afisare();
    }
}

int main() {
    vector<Vehicul*> vehicule;
    citireFisier(vehicule, "parcare.txt");
    afisareVehicule(vehicule);
    for (auto& vehicul : vehicule) delete vehicul;
    return 0;
}
```