Laborator POO : Car Race Simulator

322ABb

26/11/2024

1 Introducere

Pasionat de curse de mașini și de tehnologie, studentul nostru Mircea s-a gândit să dezvolte o aplicație care să simuleze o cursă de mașini virtuală. Inspirat de marile competiții auto, Mircea a conceput un sistem care să modeleze diferite tipuri de mașini, fiecare cu caracteristici unice, și un circuit în care acestea concurează.

În cadrul acestui laborator, vă propunem să îl ajutați să finalizeze acest proiect interesant, aplicând conceptele de polimorfism, moștenire, metode virtuale și organizarea unui proiect modularizat.

2 Structura proiectului

Proiectul este structurat în mai multe clase:

- **Car**: Clasa abstractă de bază care definește caracteristicile generale ale unei mașini.
  - Metode virtual pure: `GoCar()` și `getName()`.
  - Atribute protejate: `fuelCapacity`, `fuelConsumption`, `averageSpeed[3]`, `name`.
- **Mazda, Dacia, Toyota, Ford, Skoda (sau altele)**: Clase derivate care implementează metodele virtuale ale clasei `Car` (maxim 5).
- **Weather**: Clasă care definește condițiile meteorologice.
- **Circuit**: Clasă care gestionează configurarea cursei, adăugarea mașinilor și simularea cursei.
  - Metode: `SetLength()`, `SetWeather()`, `AddCar()`, `Race()`, `ShowFinalRanks()`, `ShowWhoDidNotFinish()`.

3 Vizualizarea ierarhiei claselor

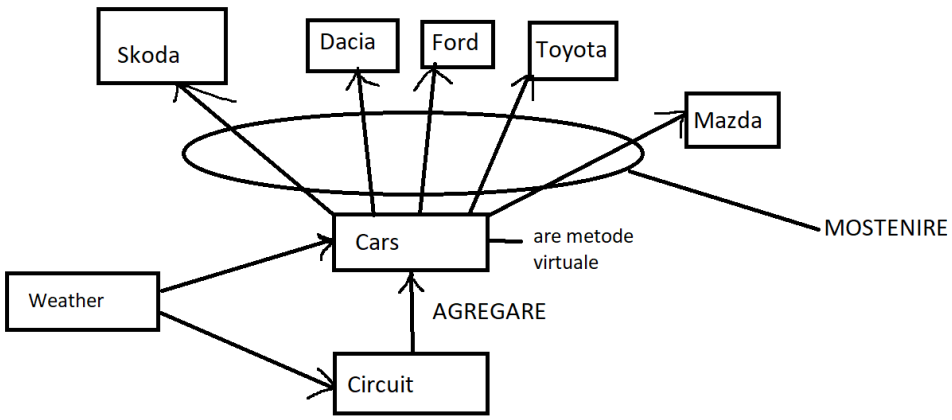

Figure 1: Schema claselor proiectului (cu exemple de clase derivate)

4 Cod sursă

Aici am pus pentru a vizualiza mai bine codul, însă veți primi, pe lângă acest document, încă 2 fișiere numite `Laborator 9 - Headers.txt` și `Laborator 9 - Sources.txt` ce vor avea fișierele header necesare pentru început, respectiv fișierele sursă pentru exemplul de implementare al derivatei, circuitului si main-ului.

4.1 ProjectLibs.h

```
#ifndef PROJECTLIBS_H_INCLUDED
#define PROJECTLIBS_H_INCLUDED

#include <iostream>
#include <cstring>

#include "Weather.h"

// Daca este necesar, mai includeti si alte biblioteci aici

#endif // PROJECTLIBS_H_INCLUDED
```

4.2 Weather.h

```
#ifndef WEATHER_H_INCLUDED
#define WEATHER_H_INCLUDED

enum Weather {Rain, Sunny, Snow};

#endif // WEATHER_H_INCLUDED
```

4.3 Car.h

```
#ifndef CAR_H_INCLUDED
#define CAR_H_INCLUDED

#include "ProjectLibs.h"

class Car {
protected:
    int fuelCapacity;
    int fuelConsumption;
    int averageSpeed[3];
    char* name;

public:
    // TODO: Scrieti destructorul acestei clase care sa fie virtual pentru a evita memory leak
        // OBS: Aveti grija ca nu este pur virtual deci trebuie implementat in .cpp

    // TODO: Modificati aceste metode astfel incat ele sa fie virtuale pure
    float GoCar(bool&, int, int) = 0;
    char* getName() = 0;
};

#endif // CAR_H_INCLUDED
```

4.4 Mazda.h

```
#ifndef MAZDA_H_INCLUDED
#define MAZDA_H_INCLUDED

#include "Car.h"

class Mazda : public Car {
public:
    Mazda();

    float GoCar(bool& b, int w, int circuitLength) override;
    char* getName() override;
};

#endif // MAZDA_H_INCLUDED
```

4.5 Mazda.cpp (model si pentru celelalte derivate)

```cpp
#include "Mazda.h"

Mazda::Mazda() {
    fuelCapacity = 100;
    fuelConsumption = 6;
    averageSpeed[Rain] = 55;
    averageSpeed[Sunny] = 100;
    averageSpeed[Snow] = 40;
    name = (char*)"Mazda";
}

float Mazda::GoCar(bool& b, int w, int circuitLength) {
    int speed = averageSpeed[w];
    float hours = fuelCapacity / fuelConsumption;
    float distance = hours * speed;

    b = distance > circuitLength;
    return (hours * circuitLength / distance);
}

char* Mazda::getName() {
    return name;
}
```

4.6 Circuit.h

```cpp
#ifndef CIRCUIT_H_INCLUDED
#define CIRCUIT_H_INCLUDED

#include "ProjectLibs.h"
#include "Mazda.h"
// TODO: Adaugati alte clase precum Dacia, Ford, Toyota, etc.

class Circuit {
private:
    int circuitLength;
    Car* car[10];
    float timeToFinish[10];
    bool finish[10];
    int cars;
    int weather;

public:
    Circuit();
    void SetLength(int);
    void SetWeather(int);
    void AddCar(Car* c);
    void Race();
    void ShowFinalRanks();
    void ShowWhoDidNotFinish();
};

#endif // CIRCUIT_H_INCLUDED
```

4.7 Circuit.cpp

```cpp
// Circuit.cpp
#include "Circuit.h"

// TODO: Initializati atributele clasei Circuit.
Circuit::Circuit() {
     // Initializati lungimea circuitului cu 0.
     // Initializati vectorul de masini, fara elemente in el
     // Initializati vectorul pentru timpii de finalizare, fara elemente in el
     // Initializati vectorul care va seta finalitatea masinilor, fara elemente in el
     // Initializati numarul de masini la 0 (este index pentru vectorul car).
     // Setati vremea implicita la Sunny.
}

// TODO: Implementati metoda pentru setarea lungimii circuitului.


// TODO: Implementati metoda pentru setarea vremii.
// 0 - Ploaie, 1 - Soare, 2 - Zapada.

void Circuit::AddCar(Car* c) {
    // TODO: Adaugati masina la vectorul de masini al circuitului si incrementati indexul.
        // OBS: Nu uitati sa verificati daca se depaseste numarul default de masini setat (10).
}

// Logica cursei
void Circuit::Race() {
    for (int i = 0; i < cars; i++) {
        // Apelam metoda GoCar pentru fiecare masina.
        bool canFinish = false;

        // Calculam timpii necesari pentru a termina cursa.
        timeToFinish[i] = car[i]->GoCar(canFinish, weather, circuitLength);
        finish[i] = canFinish;
    }
}

// TODO: Afisati clasamentul final al masinilor.
void Circuit::ShowFinalRanks() {
    // Afisam doar masinile care au terminat cursa
}

// TODO: Afisati masinile care nu au terminat cursa.
void Circuit::ShowWhoDidNotFinish() {
    // Asemanator cu ShowFinalRanks()
}
```

4.8 main.cpp

```
#include "Circuit.h"
#include "Car.h"

int main() {
    Circuit c;
    c.SetLength(1000);
    c.SetWeather(Sunny);

    Mazda* mazda = new Mazda();
    // TODO: Crearea altor clase de masini

    c.AddCar(mazda);
    // TODO: Adaugati alte masini

    c.Race();
    c.ShowFinalRanks();
    c.ShowWhoDidNotFinish();

    return 0;
}
```

5 Punctaj

- Implementare clasa abstracta: 4p
- Implementare clase derivate și metode suprascrise: 3p
- Simularea cursei și afișarea rezultatelor: 2p
- Organizare cod și utilizarea resurselor: 1p