Министерство образования Республики Беларусь

Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»

Кафедра электронных вычислительных машин

ОТЧЕТ

О ЛАБОРАТОРНОЙ РАБОТЕ № 4

Виртуальные функции. Абстрактные классы

по дисциплине «Программирование на языках высокого уровня»

Выполнил ст. гр. 450503 А.П. Красько

Проверил асс. каф. ЭВМ И.Г. Скиба

Минск 2025

# 1 ПОСТАНОВКА ЗАДАЧИ

Создать абстрактный базовый класс «транспортное средство» и производные классы «Автомобиль», «Велосипед», «Повозка». Подсчитать время и стоимость перевозки пассажиров и грузов каждым транспортным средством.

# 2 ДИАГРАММА КЛАССОВ

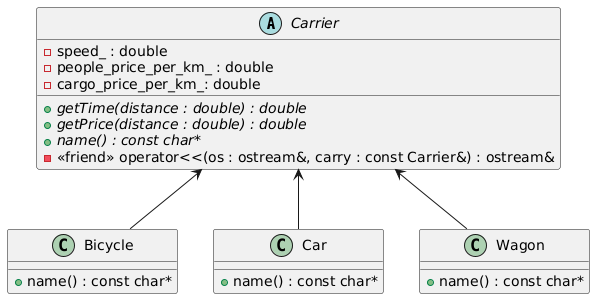


Рисунок 2.1 – Диаграмма классов

# 3 ЛИСТИНГ КОДА

Файл carrier.hh:

#pragma once  
#include <cmath>  
#include <format>  
#include <iostream>  
#include <stdexcept>  
  
namespace carriers {  
class Carrier {  
 private:  
 double speed\_;  
 double people\_price\_per\_km\_;  
 double cargo\_price\_per\_km\_;  
  
 private:  
 friend std::ostream& operator<<(std::ostream& os, const Carrier& carry) {  
 os << std::format("{}: [speed: {}, price for people: {} price for cargo: {}]", carry.name(), carry.speed\_,  
 carry.people\_price\_per\_km\_, carry.cargo\_price\_per\_km\_)  
 << std::endl;  
 return os;  
 }  
  
 public:  
 Carrier(double speed, double people\_price\_per\_km\_, double cargo\_price\_per\_km\_);  
 virtual double getTime(double distance) const;  
 virtual double getPriceForPeople(double distance) const;  
 virtual double getPriceForCargo(double distance) const;  
 virtual const char\* name() const = 0;  
};  
  
} // namespace carriers

Файл car.hh:

#pragma once  
#include "carrier.hh"  
  
namespace carriers {  
class Car : public Carrier {  
 public:  
 Car(double speed, double people\_price\_per\_km\_, double cargo\_price\_per\_km\_);  
 const char\* name() const override;  
};  
} // namespace carriers

Файл screens.hh:

#pragma once  
#include <memory>  
#include "carrier.hh"  
namespace screens {  
void printMainScreen();  
  
bool createCarrier(std::unique\_ptr<carriers::Carrier> &carry\_ptr);  
bool printCarrier(const carriers::Carrier \*carry\_ptr);  
bool calculate(const carriers::Carrier \*carry\_ptr);  
} // namespace screens

Файл wagon.hh:

#pragma once  
#include "carrier.hh"  
  
namespace carriers {  
class Wagon : public Carrier {  
 public:  
 Wagon(double speed, double people\_price\_per\_km\_, double cargo\_price\_per\_km\_);  
 const char\* name() const override;  
};  
} // namespace carriers

Файл bicycle.hh:

#pragma once  
#include "carrier.hh"  
  
namespace carriers {  
class Bicycle : public Carrier {  
 public:  
 Bicycle(double speed, double people\_price\_per\_km\_, double cargo\_price\_per\_km\_);  
 const char\* name() const override;  
};  
} // namespace carriers

Файл carrier.cc:

#include <l4/include/carrier.hh>  
namespace carriers {  
  
Carrier::Carrier(double speed, double people\_price\_per\_km\_, double cargo\_price\_per\_km\_)  
 : speed\_{speed}, people\_price\_per\_km\_{people\_price\_per\_km\_}, cargo\_price\_per\_km\_{cargo\_price\_per\_km\_} {  
 if (speed\_ <= 0) throw std::invalid\_argument("speed should be > 0");  
}  
  
double Carrier::getTime(double distance) const { return distance / speed\_; }  
double Carrier::getPriceForPeople(double distance) const { return distance \* people\_price\_per\_km\_; }  
double Carrier::getPriceForCargo(double distance) const { return distance \* cargo\_price\_per\_km\_; }  
  
} // namespace carriers

Файл bicycle.cc:

#include <l4/include/bicycle.hh>  
  
namespace carriers  
{  
 Bicycle::Bicycle(double speed, double people\_price\_per\_km\_, double cargo\_price\_per\_km\_) : Carrier(speed, people\_price\_per\_km\_, cargo\_price\_per\_km\_) {};  
 const char\* Bicycle::name() const { return "Bicycle"; }  
} // namespace carriers

Файл screens.cc:

#include <consoleUtils.hh>  
#include <l4/include/bicycle.hh>  
#include <l4/include/car.hh>  
#include <l4/include/wagon.hh>  
#include <memory>  
#include <print>  
using namespace std;  
using namespace carriers;  
using namespace console\_utils;  
  
namespace screens {  
void printMainScreen() {  
 auto [cols, rows] = getConsoleDimensions();  
 println("{:^{}}", "\x{1B}[48;5;35mLab 4\x{1B}[0m", cols);  
 println("Please select action:\n");  
 println(" 1.Create carrier");  
 println(" 2.Print carrier");  
 println(" 3.Calculate");  
 println(" 4.Exit");  
}  
  
bool createCarrier(unique\_ptr<Carrier> &carry\_ptr) {  
 unsigned int response;  
 double speed;  
 double cost\_p;  
 double cost\_c;  
 println("What type of carrier to create?");  
 println(" 1. Wagon");  
 println(" 2. Car");  
 println(" 3. Bicycle");  
 readT(response, ">", [](unsigned int num) { return num > 0 && num <= 3; });  
 readT(speed, "Please enter speed (speed > 0): ", [](double num) { return num > 0; });  
 readT(cost\_p, "Please enter cost per km for people: ");  
 readT(cost\_c, "Please enter cost per km for cargo: ");  
  
  
 switch (response) {  
 case 1:  
 carry\_ptr = make\_unique<Wagon>(speed, cost\_p, cost\_c);  
 break;  
 case 2:  
 carry\_ptr = make\_unique<Car>(speed, cost\_p, cost\_c);  
 break;  
 case 3:  
 carry\_ptr = make\_unique<Bicycle>(speed, cost\_p, cost\_c);  
 break;  
 default:  
 break;  
 }  
 return true;  
}  
bool printCarrier(const Carrier \*carry\_ptr) {  
 if (!carry\_ptr) {  
 cout << "None, please create one first" << endl;  
 return true;  
 }  
 cout << \*carry\_ptr;  
 return true;  
}  
bool calculate(const Carrier \*carry\_ptr) {  
 if (!carry\_ptr) {  
 cout << "No carrier, please create one first" << endl;  
 return true;  
 }  
 size\_t distance;  
 readT(distance, "Pleaes enter distance: ");  
 cout << format("time: {:.2f} price for people: {:.2f} price for cargo: {:.2f}", carry\_ptr->getTime(distance),  
 carry\_ptr->getPriceForPeople(distance), carry\_ptr->getPriceForCargo(distance))  
 << endl;  
  
 return true;  
}  
} // namespace screens

Файл main.cc:

#include <consoleUtils.hh>  
#include <functional>  
#include <l3/include/airplane.hh>  
#include <l3/include/car.hh>  
#include <l3/include/screens.hh>  
#include <l3/include/train.hh>  
#include <memory>  
  
using namespace std;  
using namespace carriers;  
using namespace console\_utils;  
using namespace screens;  
  
int main(void) {  
 unique\_ptr<Carrier> carry\_ptr;  
 static array<function<bool()>, 4> actions = {  
 [&carry\_ptr]() { return createCarrier(carry\_ptr); },  
 [&carry\_ptr]() { return printCarrier(carry\_ptr.get()); },  
 [&carry\_ptr]() { return calculate(carry\_ptr.get()); },  
 []() { return false; },  
  
 };  
  
 unsigned int response;  
 do {  
 printMainScreen();  
 readT(response, ">", [](unsigned int num) { return num > 0 && num <= 4; });  
 cout << "\x{1B}[2J\x{1B}[H\n";  
 } while (actions[response - 1]());  
  
 return 0;  
}

Файл car.cc:

#include <l4/include/car.hh>  
  
namespace carriers  
{  
 Car::Car(double speed, double people\_price\_per\_km\_, double cargo\_price\_per\_km\_) : Carrier(speed, people\_price\_per\_km\_, cargo\_price\_per\_km\_) {};  
 const char\* Car::name() const { return "Car"; }  
} // namespace carriers

Файл wagon.cc:

#include <l4/include/wagon.hh>  
  
namespace carriers  
{  
 Wagon::Wagon(double speed, double people\_price\_per\_km\_, double cargo\_price\_per\_km\_) : Carrier(speed, people\_price\_per\_km\_, cargo\_price\_per\_km\_) {};  
 const char\* Wagon::name() const { return "Bicycle"; }  
} // namespace carriers

**4 РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ**

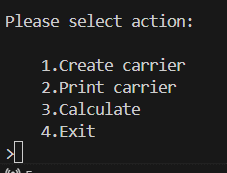


Рисунок 4.1 – Главное меню

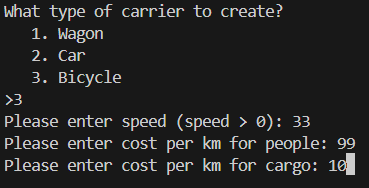


Рисунок 4.2 – Меню создания перевозчика

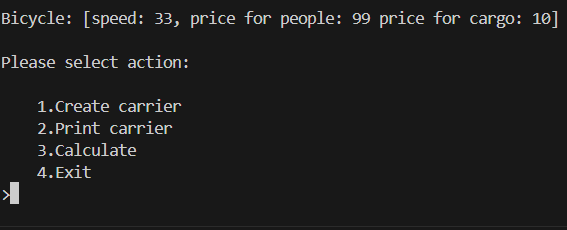


Рисунок 4.3 – Вывод перевозчика

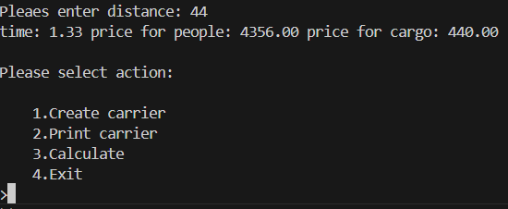


Рисунок 4.4 – Расчёт времени и цены

**5 ЗАКЛЮЧЕНИЕ**

В ходе выполнения лабораторной работы была успешно реализована иерархия классов на основе абстрактного базового класса «транспортное средство». Использование виртуальных функций и механизма полиморфизма позволило корректно организовать расчёт времени и стоимости перевозки для различных типов транспортных средств.