

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

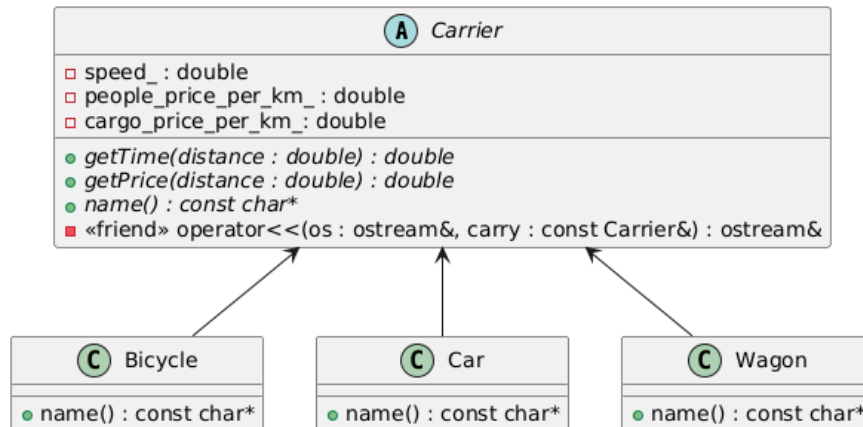


Рисунок 2.1 – Диаграмма классов

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:

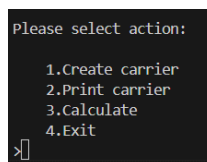
```

#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

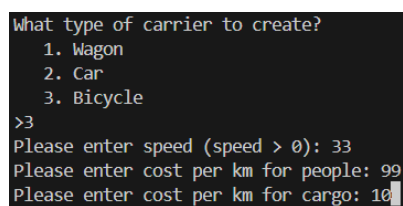


```

Please select action:
1.Create carrier
2.Print carrier
3.Calculate
4.Exit
>

```

Рисунок 4.1 – Главное меню



```

What type of carrier to create?
1. Wagon
2. Car
3. Bicycle
>3
Please enter speed (speed > 0): 33
Please enter cost per km for people: 99
Please enter cost per km for cargo: 10

```

Рисунок 4.2 – Меню создания перевозчика

```
Bicycle: [speed: 33, price for people: 99 price for cargo: 10]
Please select action:
1.Create carrier
2.Print carrier
3.Calculate
4.Exit
>
```

Рисунок 4.3 – Вывод перевозчика

```
Please enter distance: 44
time: 1.33 price for people: 4356.00 price for cargo: 440.00
Please select action:
1.Create carrier
2.Print carrier
3.Calculate
4.Exit
>
```

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.