Министерство образования и науки Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования «Алтайский государственный технический университет им. И.И. Ползунова»

Факультет информационных технологий

Кафедра прикладной математики

Отчёт защищён с оценкой\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Преподаватель Троицкий В.С.

«\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2020 г.

Отчёт

по лабораторной работе № 7

«Ассоциация»

по дисциплине «Программирование - 3»

Студент группы ПИ 92 В.М. Шульпов

Преподаватель доцент, к.т. н. Троицкий В.С.

Барнаул 2020

ЗАДАЧА

**C++**

1) Продемонстрировать возврат значения из метода через указатель (\*) и через ссылку (&);

2) Продемонстрировать разумное использование this;

3) Создать дружественную функцию и продемонстрировать ее использование;

4) Выполнить перегрузку операторов '+', '++' (два варианта, префиксный и постфиксный). Продемонстрировать в main;

5) Заменить массивы char на std::string, продемонстрировать работу с этим классом;

**Java**

6) Продемонстрировать работу с массивом объектов;

7) Продемонстрировать возврат целочисленного значения из метода через вспомогательный класс;

8) Продемонстрировать разумное использование this;

9) Продемонстрировать обработку строк (String);

**C#**

+10) Для полей добавить свойства и продемонстрировать работу с ними;

+11) Cоздать массив объектов и продемонстрировать работу с ним;

12) В отдельной ветке проекта заменить класс (class) на структуру (struct). Продемонстрировать различие между присваиванием объектов класса и структуры;

+14) Продемонстрировать возврат значения через параметр out и через параметр ref. Показать различие этих механизмов;

+16) Продемонстрировать перегрузку операторов '+', '++'.

+17) Продемонстрировать обработку строк (string);

!) Использование Git обязательно.

ТЕОРИЯ

**Когда использовать возврат по адресу**:

  при возврате динамически выделенной памяти;

  при возврате аргументов функции, которые были переданы по адресу.

**Когда не использовать возврат по адресу**:

  при возврате переменных, которые были объявлены внутри функции (используйте возврат по значению);

  при возврате большой структуры или класса, который был передан по ссылке (используйте возврат по ссылке).

**Когда использовать возврат по ссылке**:

  при возврате ссылки-параметра;

  при возврате элемента массива, который был передан в функцию;

  при возврате большой структуры или класса, который не уничтожается в конце функции (например, тот, который был передан в функцию).

**Когда не использовать возврат по ссылке**:

  при возврате переменных, которые были объявлены внутри функции (используйте возврат по значению);

  при возврате стандартного массива или значения указателя (используйте возврат по адресу).

2)

|  |  |  |
| --- | --- | --- |
| **Вопрос** | **Структура** | **Класс** |
| И какого же типа экземпляр объекта? | Структура значимый (value) тип | Класс ссылочный (reference) тип |
| А где “живут” экземпляры этих объектов? | Экземпляры структуры называют *значениями* и “живут” они в стеке (stack). | Экземпляры классов называют *объектами* и “живут” они в куче (heap). |
| Можно ли создать конструктор по умолчанию? | Нет | Да |
| Если создается свой конструктор будет ли компилятор генерировать конструктор по умолчанию? | Да | Нет |
| Если в своём конструкторе не будут инициализированы некоторые поля, будут ли они автоматически инициализированы компилятором? | Нет | Да |
| Разрешается ли инициализировать переменные в месте их объявления? | Нет | Да |

КОД ПРОГРАММЫ

1. **Класс С++**

/\*

// lab7.cpp : Этот файл содержит функцию "main". Здесь начинается и заканчивается выполнение программы.

//

#include <iostream>

class Player

{

public:

Player();

~Player();

void setPosition(int x, int y, int z);

friend void setStartPosition(Player \*player);

void displayPosition();

Player operator + (Player b); // сложение двух объектов operator +

private:

int x = 0;

int y = 0;

int z = 0;

int money = 0;

};

Player::Player()

{

}

Player::~Player()

{

}

void Player::setPosition(int x, int y, int z)

{

this->x = x;

this->y = y;

this->z = z;

}

void Player::displayPosition()

{

std::cout << "x = " << this->x << ", y = " << this->y << ", z = " << this->z << std::endl;

}

Player Player::operator+(Player b)

{

return Player();

}

void setStartPosition(Player \*player) {

player->x = 0;

player->y = 0;

player->z = 0;

}

int main()

{

//разумное использование this;

Player\* player = new Player;

player->setPosition(10, 20, 150);//в функции setPosition this указывает на тот объект, у которого мы вызываем метод

std::cout << delimiter;

// =================================== 3 ===================================

//Дружественная функция

std::cout << "параметры ДО использования дружесетвенной функции" << std::endl;

player->displayPosition();

setStartPosition(player);

std::cout << "параметры ПОСЛЕ использования дружесетвенной функции" << std::endl;

player->displayPosition();

std::cout << delimiter;

// =================================== 4 ===================================

// =================================== 5 ===================================

// =========================================================================

return 0;

}

\*/

// lab work №7 Shulpov Victor PI-92

/\*

Реализовать работу автомобиля на примере динамической структуры.

Поля структуры:

• Название (строка)

• Цена (целое)

• Цвет (строка)

• Количество оборотов двигателя в минуту (целое)

• Скорость (целое)

• Количество бензина (целое)

Функции:

• Инициализация

• Установка параметров автомобиля

• Вывод данных машины

• Запуск двигателя

• Остановка двигателя

• Добавление скорости

• Уменьшить скорость

\*/

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <malloc.h>

#include <conio.h>

#include <string.h>

#define clear(stream) rewind((stream)) //очистка потока

const int CAR\_NUMBERS = 10;

class Car;

int\* allocateArray(int size);

int& getCallNumber();

void setStartPosition(Car\* car);

class Engine

{

public:

void init(int engineRPM, int capacity, int enginePower, int quantityOfCylinders);

void setEngineRPM(int engineRPM);

void setCapacity(int capacity);

void setEnginePower(int enginePower);

void setQuantityOfCylinders(int quantityOfCylinders);

int getEngineRPM();

int getCapacity();

int getEnginePower();

int getQuantityOfCylinders();

private:

int engineRPM; //количество оборотов в минуту

int capacity; //объем см куб

int enginePower; //мощность Л.С.

int quantityOfCylinders; //количество цилиндров

};

class Car

{

public:

Car();

~Car();

void init(std::string name, int price, std::string color, int speed, int benzine, Engine\* engine);

void readCarData();

void displayDataCar();

void addBenzine(int liters);

void startEngine();

void stopEngine();

void addSpeed(int speed);

void reduceSpeed(int speed);

friend void setStartPosition(Car\* car);

Car operator+(int benzine);//перегрузка оператора +

Car& operator++ ();//префиксная

Car operator++ (int unused);

private:

int x=0;

int y=0;

std::string name;

int price;

std::string color;

int speed;

int benzine;

Engine\* engine;

};

Car::Car()

{

this->x = 0;

this->y = 0;

this->name = "";

this->price = 0;

this->color = "";

this->speed = 0;

this->benzine = 0;

this->engine = NULL;

}

Car::~Car()

{

}

void Car::init(std::string name, int price, std::string color, int speed, int benzine, Engine\* engine)

{

this->name = name;

this->price = price;

this->color = color;

this->benzine = benzine;

this->speed = speed;

this->engine = engine;

printf("Car initialized!\n");

}

void Car::readCarData() {

int number;

std::cout << "ENTER CAR DATA:" << std::endl;

std::cout << "\tname:\t";

std::cin >> this->name;

clear(stdin);

std::cout << "\tprice:\t";

std::cin >> this->price;

clear(stdin);

std::cout << "\tcolor:\t";

std::cin >> this->color;

clear(stdin);

std::cout << "\tengineRPM:\t";

std::cin >> number;

this->engine->setEngineRPM(number);

clear(stdin);

std::cout << "\tcapacity:\t";

std::cin >> number;

this->engine->setCapacity(number);

clear(stdin);

std::cout << "\tengine power:\t";

std::cin >> number;

this->engine->setEnginePower(number);

clear(stdin);

std::cout << "\tquantity of cylinders:\t";

std::cin >> number;

this->engine->setQuantityOfCylinders(number);

clear(stdin);

std::cout << "\tspeed:\t";

std::cin >> this->speed;

clear(stdin);

std::cout << "\tbenzine:\t";

std::cin >> this->benzine;

clear(stdin);

}

void Car::displayDataCar()

{

std::cout << "\t" << "Car data" << std::endl;

std::cout << "\t\tName:\t" << this->name << std::endl;

std::cout << "\t\tPrice:\t" << this->price << std::endl;

std::cout << "\t\tColor:\t" << this->color << std::endl;

std::cout << "\t\tEngineRPM:\t" << this->engine->getEngineRPM() << std::endl;

std::cout << "\t\tCapacity:\t" << this->engine->getCapacity() << std::endl;

std::cout << "\t\tEngine Power:\t" << this->engine->getEnginePower() << std::endl;

std::cout << "\t\tQuanity of cylinders:\t" << this->engine->getQuantityOfCylinders() << std::endl;

std::cout << "\t\tBenzine:\t" << this->benzine << std::endl;

std::cout << "\t\tSpeed:\t" << this->speed << std::endl;

}

void Car::addBenzine(int liters)

{

std::cout << liters << "lit. benzine added!" << std::endl;

this->benzine += liters;

}

void Car::startEngine()

{

if (this->benzine > 0) {

this->engine->setEngineRPM(800);

std::cout << "Engine started!" << std::endl;

}

else {

std::cout << "No benzine. Engine didn't start!" << std::endl;

}

}

void Car::stopEngine()

{

if (this->engine->getEngineRPM() > 0) {

this->engine->setEngineRPM(0);

std::cout << "Engine stopped!" << std::endl;

}

else {

std::cout << "Engine stopped already!" << std::endl;

}

}

void Car::addSpeed(int speed)

{

if (this->engine->getEngineRPM() > 0) {

this->speed += speed;

std::cout << "Car speeded up!" << std::endl;

}

else {

std::cout << "Engine isn't starting. Car didn't speed up!" << std::endl;

}

}

void Car::reduceSpeed(int speed)

{

if (this->speed > 0) {

this->speed -= speed;

std::cout << "Car speeded down!" << std::endl;

}

else {

std::cout << "Car is parking. Car didn't speed down!" << std::endl;

}

}

Car Car::operator+(int benzine)

{

Car new\_car = \*this;

new\_car.benzine = this->benzine + benzine;

return new\_car;

}

Car& Car::operator++()

{

this->benzine++;

return \*this;

}

Car Car::operator++(int unused)

{

Car car = \*this;

++this->benzine;

return car;

}

void Engine::init(int engineRPM, int capacity, int enginePower, int quantityOfCylinders)

{

this->engineRPM = engineRPM;

this->capacity = capacity;

this->enginePower = enginePower;

this->quantityOfCylinders = quantityOfCylinders;

}

void Engine::setEngineRPM(int engineRPM)

{

this->engineRPM = engineRPM;

}

void Engine::setCapacity(int capacity)

{

this->capacity = capacity;

}

void Engine::setEnginePower(int enginePower)

{

this->enginePower = enginePower;

}

void Engine::setQuantityOfCylinders(int quantityOfCylinders)

{

this->quantityOfCylinders = quantityOfCylinders;

}

int Engine::getEngineRPM()

{

return this->engineRPM;

}

int Engine::getCapacity()

{

return this->capacity;

}

int Engine::getEnginePower()

{

return this->enginePower;

}

int Engine::getQuantityOfCylinders()

{

return this->quantityOfCylinders;

}

int main()

{

setlocale(LC\_ALL, "Russian");

int choice=1;

while (choice!=0){

std::cout << "\n\nВведите 1 - ПОКАЗАТЬ 4 ЛАБУ\n" << "Введите 2 - ПОКАЗАТЬ 7 ЛАБУ\n" << "Введите 0 - ВЫХОД\n" << "ваш выбор: ";

std::cin >> choice;

std::cout << std::endl;

if (choice == 0) { break; }

if (choice == 1) {

//////////////////////////////////////STATIC OBJECT////////

std::cout << "\n\nSTATIC OBJECT\n\n";

Engine\* bmw\_engine = new Engine;

bmw\_engine->init(0, 4395, 625, 8);

Car bmw\_x6;

bmw\_x6.init("BMW\_X6", 3500000, "black", 0, 0, bmw\_engine); //инициализируем поля объекта

bmw\_x6.displayDataCar();

//bmw\_x6.readCarData();

bmw\_x6.displayDataCar();

bmw\_x6.startEngine(); //пытаемся завести двигатель

bmw\_x6.displayDataCar();

bmw\_x6.addBenzine(10); //добавляем бензин

bmw\_x6.displayDataCar();

bmw\_x6.startEngine(); //снова пытаемся завести двигатель

bmw\_x6.displayDataCar();

for (int i = 0; i < 4; i++) {

bmw\_x6.addSpeed(i \* 5); //добавляем скорость

bmw\_x6.displayDataCar();

}

for (int i = 0; i < 4; i++) {

bmw\_x6.reduceSpeed(i \* 5); //убавляем скорость

bmw\_x6.displayDataCar();

}

bmw\_x6.stopEngine(); //останавливаем двигатель

bmw\_x6.displayDataCar();

std::cout << "\n\nPress key to continue!\n\n";

\_getch();

std::system("cls");

////////////////////////////////////DYNAMIC OBJECT/////

std::cout << "\n\nDYNAMIC OBJECT\n\n";

Car\* audi\_a7 = new Car;

Engine\* audi\_engine = new Engine;

audi\_engine->init(0, 2995, 340, 6);

audi\_a7->init("audi", 2000000, "blue", 0, 0, audi\_engine); //инициализируем поля объекта

audi\_a7->displayDataCar();

//audi\_a7->readCarData();

audi\_a7->displayDataCar();

audi\_a7->startEngine(); //пытаемся завести двигатель

audi\_a7->displayDataCar();

audi\_a7->addBenzine(10); //добавляем бензин

audi\_a7->displayDataCar();

audi\_a7->startEngine(); //снова пытаемся завести двигатель

audi\_a7->displayDataCar();

for (int i = 0; i < 4; i++) {

audi\_a7->addSpeed(i \* 5); //добавляем скорость

audi\_a7->displayDataCar();

}

for (int i = 0; i < 4; i++) {

audi\_a7->reduceSpeed(i \* 5); //убавляем скорость

audi\_a7->displayDataCar();

}

audi\_a7->stopEngine(); //останавливаем двигатель

audi\_a7->displayDataCar();

delete audi\_a7;

///////////////////////////////////DYNAMIC OBJECTS ARRAY///

Car\* dynamic\_obj\_cars;

dynamic\_obj\_cars = new Car[5];

delete[] dynamic\_obj\_cars;

///////////////////////////////////MALLOC ///

Car\* dynamic\_obj\_car2;

dynamic\_obj\_car2 = (Car\*)malloc(CAR\_NUMBERS \* sizeof(Car));

free(dynamic\_obj\_car2);

//////////////////////////////////CALLOC /////

Car\* dynamic\_obj\_car3;

dynamic\_obj\_car3 = (Car\*)calloc(CAR\_NUMBERS, sizeof(Car));

free(dynamic\_obj\_car3);

//////////////////////////////////REALOC - change current allocated memory

Car\* dynamic\_obj\_car4 = (Car\*)calloc(2, sizeof(Car));

Car\* tmp;

if (dynamic\_obj\_car4 != NULL)

{

tmp = (Car\*)realloc(dynamic\_obj\_car4, CAR\_NUMBERS \* sizeof(Car));

if (tmp != NULL)

{

dynamic\_obj\_car4 = tmp;

}

free(dynamic\_obj\_car4);

}

}

if (choice == 2) {

//возврат значения из метода через указатель(\*)

int\* myArray = allocateArray(20);

// делаем что-нибудь с array

delete[] myArray;

//возврат значения из метода через ссылку(&)

std::cout << "Количество вызовов функции: " << getCallNumber() << "\n";

//перегрузка

Engine\* bmw\_engine = new Engine;

bmw\_engine->init(0, 4395, 625, 8);

Car bmw\_x6;

bmw\_x6.init("BMW\_X6", 3500000, "black", 0, 10, bmw\_engine); //инициализируем поля объекта

bmw\_x6.displayDataCar();

bmw\_x6 = bmw\_x6 + 10;//перегрузка оператора + (добавляем бензин)

bmw\_x6.displayDataCar();

bmw\_x6 = ++bmw\_x6; //(добавляем бензин)

bmw\_x6.displayDataCar();

bmw\_x6 = bmw\_x6++; //(добавляем бензин)

bmw\_x6.displayDataCar();

}

}

std::cout << "Вы вышли\n";

}

void setStartPosition(Car\* car)

{

car->x = 0;

car->y = 0;

}

int& getCallNumber()

// Возвращаем ссылку на индекс элемента массива

{

static int number;

if (number == NULL) number = 1;

else number++;

return number;

}

int\* allocateArray(int size)

//функция выделения памяти (возврат значения по адресу(указателю))

{

return new int[size];

}

1. **Структура C#**

using System;

namespace lab7

{

/////////////////////////////////////////////////

class Program

{

static void Main(string[] args)

{

//Console.WriteLine("Kek lol program is working");

Car bmw = new Car();

bmw.init("BMW X6", 3500000, "BLACK", 0, 0); //инициализируем поля объекта

bmw.displayDataCar();

bmw.readCarData();

bmw.displayDataCar();

Console.ReadKey();

}

}

///////////////////////////////////////////////

struct Car

{

private String name;

private int price;

private String color;

private int speed;

private int benzine;

/\*

Ref параметры предназначены для данных, которые могут быть изменены,

out параметры предназначены для данных, которые являются дополнительным выходом для функции

(например, int.TryParse), которые уже используют возвращаемое значение для чего-то.

\*/

public void getStr(out String str)

{

str = "Машина " + name + ", цвет " + color + ", цена " + price;

}

public void changeStr(ref String str)

{

str = str + "!!!";

}

public String secondColor { private set; get; }

public void init(String name, int price, String color, int speed, int benzine)

{

this.name = name;

this.price = price;

this.color = color;

this.secondColor = color;

this.speed = speed;

this.benzine = benzine;

Console.WriteLine("Car initialized!");

}

public void readCarData()

{

string str = "";

int number = 0;

Console.WriteLine("ENTER CAR DATA:");

//////////////////////////////////////////////////

Console.Write("\tname:\t");

this.name = Console.ReadLine();

///////////////////////////////////////////////

bool flag = false; //флаг правильности ввода чисел

while (!flag)

{

Console.Write("\tprice:\t");

str = Console.ReadLine();

flag = int.TryParse(str, out number);

if (!flag)

{

Console.Write("re-enter:");

}

}

this.price = number;

//////////////////////////////////////////////

Console.Write("\tcolor:\t");

this.color = Console.ReadLine();

///////////////////////////////////////////////////////

flag = false;

while (!flag)

{

Console.Write("\tspeed:\t");

str = Console.ReadLine();

flag = int.TryParse(str, out number);

if (!flag)

{

Console.Write("re-enter:");

}

}

this.speed = number;

///////////////////////////////////////////////////////

flag = false;

while (!flag)

{

Console.Write("\tbenzine:\t");

str = Console.ReadLine();

flag = int.TryParse(str, out number);

if (!flag)

{

Console.Write("re-enter:");

}

}

this.benzine = number;

///////////////////////////////////////////////////////

}

public void displayDataCar()

{

Console.WriteLine("\n\tCAR DATA");

Console.WriteLine("\t\tName:\t" + this.name.ToLower());

Console.WriteLine("\t\tPrice:\t" + this.price);

Console.WriteLine("\t\tColor:\t" + this.color.ToLower());

Console.WriteLine("\t\tBenzine:\t" + this.benzine);

Console.WriteLine("\t\tSpeed:\t" + this.speed);

}

public void addBenzine(int liters)

{

Console.WriteLine(liters + "lit. benzine added!");

this.benzine += liters;

}

public void addSpeed(int speed)

{

this.speed += speed;

Console.WriteLine("Car speeded up!");

}

public void reduceSpeed(int speed)

{

if (this.speed > 0)

{

this.speed -= speed;

Console.WriteLine("Car speeded down!");

}

else

{

Console.WriteLine("Car is parking. Car didn't speed down!");

}

}

}

/////////////////////////////////////////////////

}

1. **Класс C#**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace lab7

{

/////////////////////////////////////////////////

class Program

{

static void Main(string[] args)

{

//Console.WriteLine("Kek lol program is working");

Car bmw = new Car();

Car audi = new Car();

Engine engine1 = new Engine();

Engine engine2 = new Engine();

engine1.init(0, 4395, 625, 8);

engine2.init(0, 4395, 625, 8);

Car[] car = new Car[2];**//массив объектов**

bmw.init("BMW X6", 3500000, "BLACK", 0, 0, engine1); //инициализируем поля объекта

audi.init("Audi", 3300000, "BLACK", 0, 0, engine2); //инициализируем поля объекта

audi.readCarData();

car[0] = bmw;

car[1] = audi;

car[0].displayDataCar();

car[1].displayDataCar();

car[1] = car[1] + 50; **//перегрузка +**

car[1].displayDataCar();

car[1]++; **//перегрузка ++**

car[1].displayDataCar();

Console.WriteLine("\tВторой цвет: " + car[0].secondColor);

String str = "";

car[1].getStr(out str) **//out**

Console.WriteLine("out str="+str);

car[1].changeStr(ref str); **//ref**

Console.WriteLine("ref str=" + str);

Console.ReadKey();

}

}

/////////////////////////////////////////////////

class Engine

{

private int engineRPM; //количество оборотов в минуту

private int capacity; //объем см куб

private int enginePower; //мощность Л.С.

private int quantityOfCylinders; //количество цилиндров

public void init(int engineRPM, int capacity, int enginePower, int quantityOfCylinders)

{

this.engineRPM = engineRPM;

this.capacity = capacity;

this.enginePower = enginePower;

this.quantityOfCylinders = quantityOfCylinders;

}

public void setEngineRPM(int engineRPM)

{

this.engineRPM = engineRPM;

}

public void setCapacity(int capacity)

{

this.capacity = capacity;

}

public void setEnginePower(int enginePower)

{

this.enginePower = enginePower;

}

public void setQuantityOfCylinders(int quantityOfCylinders)

{

this.quantityOfCylinders = quantityOfCylinders;

}

public int getEngineRPM()

{

return this.engineRPM;

}

public int getCapacity()

{

return this.capacity;

}

public int getEnginePower()

{

return this.enginePower;

}

public int getQuantityOfCylinders()

{

return this.quantityOfCylinders;

}

}

/////////////////////////////////////////////////

class Car

{

private String name; **//String тип**

private int price;

private String color; **//String тип**

private int speed;

private int benzine;

private Engine engine;

/\*

Ref параметры предназначены для данных, которые могут быть изменены,

out параметры предназначены для данных, которые являются дополнительным выходом для функции

(например, int.TryParse), которые уже используют возвращаемое значение для чего-то.

\*/

public void getStr(out String str) **//out**

{

str = "Машина " + name + ", цвет " + color + ", цена " + price;

}

public void changeStr(ref String str)

{

str = str + "!!!";**//ref**

}

public String secondColor { private set; get; }**//свойства полей**

public void init(String name, int price, String color, int speed, int benzine, Engine engine)

{

this.name = name;

this.price = price;

this.color = color;

this.secondColor = color;

this.speed = speed;

this.benzine = benzine;

this.engine = engine;

Console.WriteLine("Car initialized!");

}

public void readCarData()

{

string str = "";

int number = 0;

Console.WriteLine("ENTER CAR DATA:");

//////////////////////////////////////////////////

Console.Write("\tname:\t");

this.name = Console.ReadLine();

///////////////////////////////////////////////

bool flag = false; //флаг правильности ввода чисел

while (!flag)

{

Console.Write("\tprice:\t");

str = Console.ReadLine();

flag = int.TryParse(str, out number);

if (!flag)

{

Console.Write("re-enter:");

}

}

this.price = number;

//////////////////////////////////////////////

Console.Write("\tcolor:\t");

this.color = Console.ReadLine();

//////////////////////////////////////////////////

flag = false;

while (!flag)

{

Console.Write("\tengineRPM:\t");

str = Console.ReadLine();

flag = int.TryParse(str, out number);

if (!flag)

{

Console.Write("re-enter:");

}

}

this.engine.setEngineRPM(number);

////////////////////////////////////////////////////

flag = false;

while (!flag)

{

Console.Write("\tcapacity:\t");

str = Console.ReadLine();

flag = int.TryParse(str, out number);

if (!flag)

{

Console.Write("re-enter:");

}

}

this.engine.setCapacity(number);

///////////////////////////////////////////////////////

flag = false;

while (!flag)

{

Console.Write("\tengine power:\t");

str = Console.ReadLine();

flag = int.TryParse(str, out number);

if (!flag)

{

Console.Write("re-enter:");

}

}

this.engine.setEnginePower(number);

///////////////////////////////////////////////////////

flag = false;

while (!flag)

{

Console.Write("\tquantity of cylinders:\t");

str = Console.ReadLine();

flag = int.TryParse(str, out number); **//out**

if (!flag)

{

Console.Write("re-enter:");

}

}

this.engine.setQuantityOfCylinders(number);

///////////////////////////////////////////////////////

flag = false;

while (!flag)

{

Console.Write("\tspeed:\t");

str = Console.ReadLine();

flag = int.TryParse(str, out number);

if (!flag)

{

Console.Write("re-enter:");

}

}

this.speed = number;

///////////////////////////////////////////////////////

flag = false;

while (!flag)

{

Console.Write("\tbenzine:\t");

str = Console.ReadLine();

flag = int.TryParse(str, out number);

if (!flag)

{

Console.Write("re-enter:");

}

}

this.benzine = number;

///////////////////////////////////////////////////////

}

public void displayDataCar()

{

Console.WriteLine("\n\tCAR DATA");

Console.WriteLine("\t\tName:\t" + this.name.ToLower());**//обработка строки**

Console.WriteLine("\t\tPrice:\t" + this.price);

Console.WriteLine("\t\tColor:\t" + this.color.ToLower());**//обработка строки**

Console.WriteLine("\t\tEngineRPM:\t" + this.engine.getEngineRPM());

Console.WriteLine("\t\tCapacity:\t" + this.engine.getCapacity());

Console.WriteLine("\t\tEngine Power:\t" + this.engine.getEnginePower());

Console.WriteLine("\t\tQuanity of cylinders:\t" + this.engine.getQuantityOfCylinders());

Console.WriteLine("\t\tBenzine:\t" + this.benzine);

Console.WriteLine("\t\tSpeed:\t" + this.speed);

}

public void addBenzine(int liters)

{

Console.WriteLine(liters + "lit. benzine added!");

this.benzine += liters;

}

public void startEngine()

{

if (this.benzine > 0)

{

this.engine.setEngineRPM(800);

Console.WriteLine("Engine started!");

}

else

{

Console.WriteLine("No benzine. Engine didn't start!");

}

}

public void stopEngine()

{

if (this.engine.getEngineRPM() > 0)

{

this.engine.setEngineRPM(0);

Console.WriteLine("Engine stopped!");

}

else

{

Console.WriteLine("Engine stopped already!");

}

}

public void addSpeed(int speed)

{

if (this.engine.getEngineRPM() > 0)

{

this.speed += speed;

Console.WriteLine("Car speeded up!");

}

else

{

Console.WriteLine("Engine isn't starting. Car didn't speed up!");

}

}

public void reduceSpeed(int speed)

{

if (this.speed > 0)

{

this.speed -= speed;

Console.WriteLine("Car speeded down!");

}

else

{

Console.WriteLine("Car is parking. Car didn't speed down!");

}

}

public static Car operator +(Car car, int benzine) **//перегрузка +**

{

car.benzine += benzine;

return car;

}

public static Car operator ++(Car car) // ++ один префикс и постфикс!

{

++car.benzine; **//перегрузка ++**

return car;

}

}

/////////////////////////////////////////////////

}

1. **Java class**

package com.company;

import java.io.\*;// подключение библиотеки ввода-вывода на java

import java.util.Scanner;//консольный ввод

/\*

Поля класса:

• Название (строка)

• Цена (целое)

• Цвет (строка)

• Количество оборотов двигателя в минуту (целое)

• Скорость (целое)

• Количество бензина (целое)

Функции:

• Инициализация

• Установка параметров автомобиля

• Вывод данных машины

• Запуск двигателя

• Остановка двигателя

• Добавление скорости

• Уменьшить скорость

\*/

public class Main {

public static void main(String[] args) {

try{

Car[] car = new Car[3];**//МАССИВ ОБЪЕКТОВ**

car[0] = new Car();

Engine bmw\_engine = new Engine();

bmw\_engine.init(0, 4395, 625, 8);

car[0].init("BMW X6", 3500000, "BLACK", 0, 22, bmw\_engine); //инициализируем поля объекта

car[0].displayDataCar();

GasStation gasStation = new GasStation();

**//возврат целочисленного значения из метода через вспомогательный класс;**

int liters = gasStation.getCurrentBenzine(car[0]);

System.out.print("\tлитры:\t"+liters);

}

catch(Exception е)

{

System.out.println("вызов исключения");

}

}

}

**//ДОПОЛНИТЕЛЬНЫЙ КЛАСС**

class GasStation{

public int getCurrentBenzine(Car car){

return car.addBenzine(0);

}

}

class Engine{

private int engineRPM; //количество оборотов в минуту

private int capacity; //объем см куб

private int enginePower; //мощность Л.С.

private int quantityOfCylinders; //количество цилиндров

public void init(int engineRPM, int capacity, int enginePower, int quantityOfCylinders){

this.engineRPM = engineRPM;

this.capacity = capacity;

this.enginePower = enginePower;

this.quantityOfCylinders = quantityOfCylinders;

}

public void setEngineRPM(int engineRPM){

this.engineRPM = engineRPM;

}

public void setCapacity(int capacity){

this.capacity = capacity;

}

public void setEnginePower(int enginePower){

this.enginePower = enginePower;

}

public void setQuantityOfCylinders(int quantityOfCylinders){

this.quantityOfCylinders = quantityOfCylinders;

}

public int getEngineRPM(){

return this.engineRPM;

}

public int getCapacity(){

return this.capacity;

}

public int getEnginePower(){

return this.enginePower;

}

public int getQuantityOfCylinders(){

return this.quantityOfCylinders;

}

}

class Car{

Scanner in = new Scanner(System.in/\*, "Cp866"\*/);

//BufferedReader br = new BufferedReader(new InputStreamReader(System.in, "Cp866"));

private String name;//тип String

private int price;

private String color;//тип StringB

private int speed;

private int benzine;

private Engine engine;

public void init(String name, int price, String color, int speed, int benzine, Engine engine){

this.name = name.toLowerCase();**//ОБРАБОТКА СТРОК**

this.price = price;

this.color = color.toLowerCase();**//ОБРАБОТКА СТРОК**

this.speed = speed;

this.benzine = benzine;

this.engine = engine;

System.out.println("Car initialized!");

}

public void readCarData(){

System.out.println("ENTER CAR DATA:");

System.out.print("\tname:\t");

this.name = in.nextLine().toLowerCase();**//ОБРАБОТКА СТРОК**

System.out.print("\tprice:\t");

this.price = in.nextInt();

in.nextLine();//очистка потока

System.out.print("\tcolor:\t");

this.color = in.nextLine().toLowerCase();**//ОБРАБОТКА СТРОК**

System.out.print("\tengineRPM:\t");

this.engine.setEngineRPM(in.nextInt());

in.nextLine();//очистка потока

System.out.print("\tcapacity:\t");

this.engine.setCapacity(in.nextInt());

in.nextLine();//очистка потока

System.out.print("\tengine power:\t");

this.engine.setEnginePower(in.nextInt());

in.nextLine();//очистка потока

System.out.print("\tquantity of cylinders:\t");

this.engine.setQuantityOfCylinders(in.nextInt());

in.nextLine();//очистка потока

System.out.print("\tspeed:\t");

this.speed = in.nextInt();

in.nextLine();//очистка потока

System.out.print("\tbenzine:\t");

this.benzine = in.nextInt();

in.nextLine();//очистка потока

}

public void displayDataCar(){

try{

System.out.println("\n\tCAR DATA");

System.out.println("\t\tName:\t" + this.name);

System.out.println("\t\tPrice:\t" + this.price);

System.out.println("\t\tColor:\t" + this.color);

System.out.println("\t\tEngineRPM:\t" + this.engine.getEngineRPM());

System.out.println("\t\tCapacity:\t" + this.engine.getCapacity());

System.out.println("\t\tEngine Power:\t" + this.engine.getEnginePower());

System.out.println("\t\tQuanity of cylinders:\t" + this.engine.getQuantityOfCylinders());

System.out.println("\t\tBenzine:\t" + this.benzine);

System.out.println("\t\tSpeed:\t" + this.speed);

}

catch(Exception е)

{

System.out.println("вызов исключения");

}

}

public int addBenzine(int liters){

System.out.println(liters + "lit. benzine added!");

this.benzine += liters;

return this.benzine;

}

public void startEngine(){

if (this.benzine > 0) {

this.engine.setEngineRPM(800);

System.out.println("Engine started!");

}

else {

System.out.println("No benzine. Engine didn't start!");

}

}

public void stopEngine(){

if (this.engine.getEngineRPM() > 0) {

this.engine.setEngineRPM(0);

System.out.println("Engine stopped!");

}

else {

System.out.println("Engine stopped already!");

}

}

public void addSpeed(int speed){

if (this.engine.getEngineRPM() > 0) {

this.speed += speed;

System.out.println("Car speeded up!");

}

else {

System.out.println("Engine isn't starting. Car didn't speed up!");

}

}

public void reduceSpeed(int speed){

if (this.speed > 0) {

this.speed -= speed;

System.out.println("Car speeded down!");

}

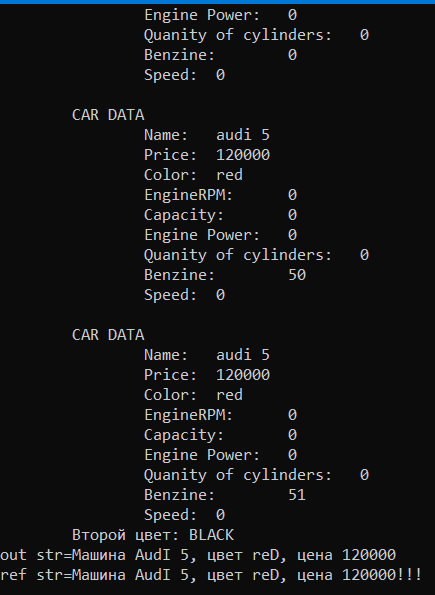
else {

System.out.println("Car is parking. Car didn't speed down!");

}

}

}

ТЕСТЫ

