

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА  
(САМАРСКИЙ УНИВЕРСИТЕТ)»

Институт Информатики и кибернетики   
Кафедра Программных систем

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**  
  
к лабораторной работе №2 по дисциплине «Прикладные информационные системы»

Обучающийся группы 6132-020402D \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Дьяконов А.В.

Обучающийся группы 6132-020402D \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Лазарев М.Ю.

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Лобанков А.А.

Самара 2024

СОДЕРЖАНИЕ

[1 Постановка задачи 3](#_Toc178189259)

[2 Результаты работы 4](#_Toc178189260)

[2.1 Результаты задания 1 4](#_Toc178189261)

[2.2 Результаты задания 2 4](#_Toc178189262)

[2.3 Результаты задания 3 5](#_Toc178189263)

[2.4 Результаты задания 4 5](#_Toc178189264)

[2.5 Результаты задания 5 6](#_Toc178189265)

[ПРИЛОЖЕНИЕ А Листинг программы 8](#_Toc178189267)

1. Постановка задачи

В процессе выполнения заданий ознакомиться с реализацией ООП и использованием модулей и пространств имён в языке TypeScript.

1. Результаты работы
   1. Результаты задания 1

Реализуйте интерфейс Владелец (Owner) со следующими свойствами:

• Фамилия

• Имя

• Отчество

• Дата рождения

• Тип документа (использовать отдельный тип перечисления)

• Серия документа

• Номер документа

• Метод, выводящий в консоль все вышеперечисленные сведения о владельце

Реализуйте интерфейс Транспортное средство (Vehicle) со следующими свойствами и определениями методов:

• Марка

• Модель

• Год выпуска

• VIN-номер

• Регистрационный номер

• Сведения о владельце

• Метод, выводящий в консоль сведения о транспортном средстве без сведений о владельце

Создайте класс, реализующий интерфейс Транспортное средство. Создайте класс, реализующий интерфейс Владелец. Каждый класс должен иметь конструктор, который имеет все необходимые параметры для инициализации объекта типа. Выведите информацию о созданных объектах. Поля внутри класса должны быть приватными. Для получения доступа к их содержимому и модификации следует использовать Геттеры и Сеттеры.

1. Создаем тип перечисления для документов enum, в котором указываем виды документов myDocType. В перечислении указываем виды документов Passport (Паспорт) и DriverLicense (Водительское удостоверение).
2. Создаем интерфейс Владелец (Owner), который описывает свойства и метод для владельца. В свойствах указываем lastName как string (Фамилия), firstName - string (Имя), middleName - string (Отчество), birthDate – Date (Дата рождения), docType – перечисление myDocType (Вид документа), docSeries - string (Серия документа) и docNumber - string (Номер документа). Метод printInfo() для вывода указанной информации.
3. Создаем интерфейс Транспортное средство (Vehicle), который описывает свойства и метод для транспортного средства. В свойствах указываем brand как string (Марка), model - string (Модель), year - number (Год выпуска), vin - string (VIN-номер), regNumber - string (Регистрационный номер) и owner – интерфейс Owner (Сведения о владельце). Метод printVehicleInfo для вывода указанной информации.
4. Создаем класс OwnerClass, который реализует интерфейс Owner и предоставляет геттеры и сеттеры для всех свойств с соответствующими приватными полями. Метод printInfo выводит информацию о владельце.
5. Создаём класс VehicleClass, который реализует интерфейс Vehicle, используя объект Owner для хранения данных о владельце и также предоставляет геттеры и сеттеры для всех свойств с соответствующими приватными полями. Метод printVehicleInfo выводит информацию о транспортном средстве без владельца.
6. Создаем объекты owner1 для класса OwnerClass и vehicle1 для класса VehicleClass, и выводим информацию (Рис. 1).

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 1 – Результат выполнения задания 1

* 1. Результаты задания 2

Создайте интерфейс Автомобиль (Car), наследующий интерфейс Транспортное средство со свойствами и определениями методов:

• Тип кузова (как перечисление)

• Класс автомобиля (как перечисление)

Создайте класс, реализующий интерфейс Автомобиль. Метод, выводящий в консоль сведения о транспортном средстве без сведений о владельце следует модифицировать для вывода дополнительных полей класса Автомобиль.

Создайте интерфейс Мотоцикл (Motorbike), наследующий интерфейс Транспортное средство со свойствами и определениями методов:

• Тип рамы (строка)

• Для спорта (булево значение)

Создайте класс, реализующий интерфейс Мотоцикл. Метод, выводящий в консоль сведения о транспортном средстве без сведений о владельце следует модифицировать для вывода дополнительных полей класса Мотоцикл.

1. Создаем перечисления enum для типа кузова CarBodyType и класса автомобиля CarClassType.
2. Создаем интерфейс Car (Автомобиль), наследующий интерфейс Транспортное средство (Vehicle). Создаем класс CarClass, реализующий данный интерфейс, добавляем приватные поля для перечислений CarBodyType и CarClassType, также геттеры и сеттеры соответственно. Переопределяем метод printVehicleInfo для класса CarClass.
3. Создаем интерфейс Motorbike (Мотоцикл), наследующий интерфейс Транспортное средство (Vehicle). Создаем класс MotorbikeClass, реализующий данный интерфейс, добавляем приватные поля для свойств значение string для frameType (Тип рамы) и булевого значения isForSport (Для спорта), также геттеры и сеттеры соответственно. Переопределяем метод printVehicleInfo для класса MotorbikeClass.
4. Наследование свойств в обоих классах осуществляем с помощью super. Создаем объекты car1 для CarClass и motorbike1 для MotorbikeClass, принадлежащие объекту owner2 для OwnerClass, и выводим информацию (Рис. 2).

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 2 – Результат выполнения задания 2

* 1. Результаты задания 3

Реализуйте интерфейс Хранилище (VehicleStorage) с обобщением типа T, ограниченное типом Транспортное средство. Хранилище содержит следующие свойства и определения методов:

• Дата создания хранилища,

• Массив T, хранящий сведения о тех или иных типах Транспортных средств,

• Метод, возвращающий все элементы массива типа Т.

Создайте класс, реализующий интерфейс Хранилище с аналогичным обобщением, как в интерфейсе.

1. Создаем интерфейс VehicleStorage с обобщением и ограничением - T extends Vehicle. Определяем свойство creationDate как Date (Дата создания хранилища) и массив vehicles: T[] (Сведения о типах транспортных средств в хранилище). Метод getVehicles(): T[] для получения всех элементов массива.
2. Создаем класс VehicleStorageClass с аналогичными обобщением и ограничением, реализующий интерфейс VehicleStorage. В constructor для свойства creationDate устанавливаем текущую дату при создании хранилища newDate(). Реализуем методы для добавления транспортного средства в хранилище addVehicle с применением к массиву vehicles.push и возврата из массива всех транспортных средств getVehicles с применением возврата return.
3. Создаем объекты car2 для CarClass и motorbike2 для MotorbikeClass, принадлежащие объекту owner3 для OwnerClass, и создаем отдельные хранилища для автомобилей carStorage и мотоциклов motorbikeStorage. Выводим все транспортные средства из хранилищ с помощью getVehicles().forEach (Рис. 3).

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 3 – Результат выполнения задания 3

* 1. Результаты задания 4

Выполните заключение интерфейсов и классов, реализованных в заданиях 1–3 в пространство имён с названием Transport. Создайте объекты с использованием данного пространства имён.

1. Создаем пространство имен namespace Transport и применяем export ко всем созданным отдельным перечислениям, классам и интерфейсам.
2. Создаем объекты с применением пространства имен. Для owner4 применяем Transport.OwnerClass, car3 – Transport.CarClass, motorbike3 – Transport.MotorbikeClass и выводим информацию (Рис. 4).

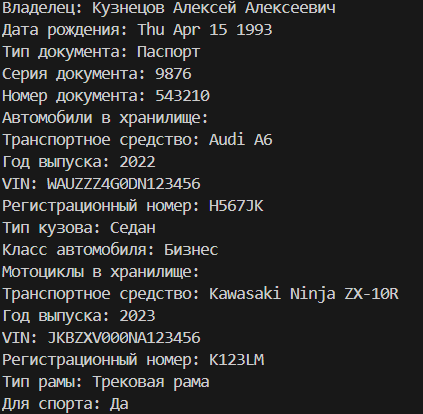


Рисунок 4 – Результат выполнения задания 4

* 1. Результаты задания 5

Добавьте ключевое слово, которое сделает namespace модулем. При необходимости добавьте это ключевое слово к другим элементам пространства имён. Выполните создание объектов и вызов методов этих объектов из другого файла.

1. Создаем файл lab2transport.ts и вписываем в него наше пространство имен, так как у нас уже в основном файле лабораторной работы имеется пространство имен Transport, называем наше пространство имен TransportforLab2 и добавляем ключевое слово export к namespace, так же export применяется ко всем классам, интерфейсам и перечислениям как в задаче 4.
2. Компилируем из полученного файла одноименный модуль js-файл при помощи команды tsc --module commonjs lab2transport.ts и применяем import … from … с указанием импорта нашего пространства имен из соответствующего js-модуля.
3. Получаем результат в соответствии с созданными объектами с вызовом из пространства имен TransportforLab2 (Рис. 5).

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 5 – Результат выполнения задания 5

ПРИЛОЖЕНИЕ А  
Листинг программы

// Задание 1

enum myDocType {

Passport = "Паспорт",

DriverLicense = "Водительское удостоверение"

}

interface Owner {

lastName: string;

firstName: string;

middleName: string;

birthDate: Date;

docType: myDocType;

docSeries: string;

docNumber: string;

printInfo(): void;

}

interface Vehicle {

brand: string;

model: string;

year: number;

vin: string;

regNumber: string;

owner: Owner;

printVehicleInfo(): void;

}

class OwnerClass implements Owner {

private \_lastName: string;

private \_firstName: string;

private \_middleName: string;

private \_birthDate: Date;

private \_docType: myDocType;

private \_docSeries: string;

private \_docNumber: string;

get lastName() {

return this.\_lastName;

}

get firstName() {

return this.\_firstName;

}

get middleName() {

return this.\_middleName;

}

get birthDate() {

return this.\_birthDate;

}

get docType() {

return this.\_docType;

}

get docSeries() {

return this.\_docSeries;

}

get docNumber() {

return this.\_docNumber;

}

set lastName(lastName: string) {

this.\_lastName = lastName;

}

set firstName(firstName: string) {

this.\_firstName = firstName;

}

set middleName(middleName: string) {

this.\_middleName = middleName;

}

set birthDate(birthDate: Date) {

this.\_birthDate = birthDate;

}

set docType(docType: myDocType) {

this.\_docType = docType;

}

set docSeries(docSeries: string) {

this.\_docSeries = docSeries;

}

set docNumber(docNumber: string) {

this.\_docNumber = docNumber;

}

constructor(

lastName: string,

firstName: string,

middleName: string,

birthDate: Date,

docType: myDocType,

docSeries: string,

docNumber: string

) {

this.\_lastName = lastName;

this.\_firstName = firstName;

this.\_middleName = middleName;

this.\_birthDate = birthDate;

this.\_docType = docType;

this.\_docSeries = docSeries;

this.\_docNumber = docNumber;

}

printInfo(): void {

console.log(`Владелец: ${this.\_lastName} ${this.\_firstName} ${this.\_middleName}`);

console.log(`Дата рождения: ${this.\_birthDate.toDateString()}`);

console.log(`Тип документа: ${this.\_docType}`);

console.log(`Серия документа: ${this.\_docSeries}`);

console.log(`Номер документа: ${this.\_docNumber}`);

}

}

class VehicleClass implements Vehicle {

private \_brand: string;

private \_model: string;

private \_year: number;

private \_vin: string;

private \_regNumber: string;

private \_owner: Owner;

get brand() {

return this.\_brand;

}

get model() {

return this.\_model;

}

get year() {

return this.\_year;

}

get vin() {

return this.\_vin;

}

get regNumber() {

return this.\_regNumber;

}

get owner() {

return this.\_owner;

}

set brand(brand: string) {

this.\_brand = brand;

}

set model(model:string) {

this.\_model = model;

}

set year(year: number) {

this.\_year = year;

}

set vin(vin: string) {

this.\_vin = vin;

}

set regNumber(regNumber: string) {

this.\_regNumber = regNumber;

}

set owner(owner: Owner) {

this.\_owner = owner;

}

constructor(

brand: string,

model: string,

year: number,

vin: string,

regNumber: string,

owner: Owner

) {

this.\_brand = brand;

this.\_model = model;

this.\_year = year;

this.\_vin = vin;

this.\_regNumber = regNumber;

this.\_owner = owner;

}

printVehicleInfo(): void {

console.log(`Транспортное средство: ${this.\_brand} ${this.\_model}`);

console.log(`Год выпуска: ${this.\_year}`);

console.log(`VIN: ${this.\_vin}`);

console.log(`Регистрационный номер: ${this.\_regNumber}`);

}

}

let owner1 = new OwnerClass(

"Лазарев",

"Михаил",

"Юрьевич",

new Date(1992, 2, 8),

myDocType.Passport,

"1234",

"567890"

);

let vehicle1 = new VehicleClass(

"Lada",

"Granta",

2013,

"JTDBE32K500123456",

"A123BC",

owner1

);

owner1.printInfo();

vehicle1.printVehicleInfo();

// Задание 2

enum CarBodyType {

Sedan = "Седан",

Hatchback = "Хэтчбек",

Crossover = "Кроссовер",

}

enum CarClassType {

Economy = "Эконом",

Business = "Бизнес",

Premium = "Премиум",

}

interface Car extends Vehicle {

carBodyType: CarBodyType;

carClassType: CarClassType;

}

class CarClass extends VehicleClass implements Car {

private \_carBodyType: CarBodyType;

private \_carClassType: CarClassType;

constructor(

brand: string,

model: string,

year: number,

vin: string,

regNumber: string,

owner: Owner,

carBodyType: CarBodyType,

carClassType: CarClassType

) {

super(brand, model, year, vin, regNumber, owner);

this.\_carBodyType = carBodyType;

this.\_carClassType = carClassType;

}

get carBodyType() {

return this.\_carBodyType;

}

get carClassType() {

return this.\_carClassType;

}

set carBodyType(carBodyType: CarBodyType) {

this.\_carBodyType = carBodyType;

}

set carClassType(carClassType: CarClassType) {

this.\_carClassType = carClassType;

}

printVehicleInfo(): void {

super.printVehicleInfo();

console.log(`Тип кузова: ${this.\_carBodyType}`);

console.log(`Класс автомобиля: ${this.\_carClassType}`);

}

}

interface Motorbike extends Vehicle {

frameType: string;

isForSport: boolean;

}

class MotorbikeClass extends VehicleClass implements Motorbike {

private \_frameType: string;

private \_isForSport: boolean;

constructor(

brand: string,

model: string,

year: number,

vin: string,

regNumber: string,

owner: Owner,

frameType: string,

isForSport: boolean

) {

super(brand, model, year, vin, regNumber, owner);

this.\_frameType = frameType;

this.\_isForSport = isForSport;

}

get frameType() {

return this.\_frameType;

}

get isForSport() {

return this.\_isForSport;

}

set frameType(frameType: string) {

this.\_frameType = frameType;

}

set isForSport(isForSport: boolean) {

this.\_isForSport = isForSport;

}

printVehicleInfo(): void {

super.printVehicleInfo();

console.log(`Тип рамы: ${this.\_frameType}`);

console.log(`Для спорта: ${this.\_isForSport ? "Да" : "Нет"}`);

}

}

let owner2 = new OwnerClass(

"Дьяконов",

"Александр",

"Викторович",

new Date(1975, 8, 15),

myDocType.DriverLicense,

"4321",

"987654"

);

let car1 = new CarClass(

"Mercedes",

"C-Class",

2019,

"WBA3A5C50FF123456",

"B456DE",

owner2,

CarBodyType.Sedan,

CarClassType.Premium

);

let motorbike1 = new MotorbikeClass(

"Ducatti",

"YZF-R1",

2021,

"JYARN23E0KA012345",

"C789GH",

owner2,

"Спортивная рама",

true

);

car1.printVehicleInfo();

motorbike1.printVehicleInfo();

// Задание 3

interface VehicleStorage<T extends Vehicle> {

creationDate: Date;

vehicles: T[];

getVehicles(): T[];

}

class VehicleStorageClass<T extends Vehicle> implements VehicleStorage<T> {

creationDate: Date;

vehicles: T[];

constructor() {

this.creationDate = new Date();

this.vehicles = [];

}

addVehicle(vehicle: T): void {

this.vehicles.push(vehicle);

}

getVehicles(): T[] {

return this.vehicles;

}

}

let owner3 = new OwnerClass(

"Сидоров",

"Сидор",

"Сидорович",

new Date(1990, 2, 28),

myDocType.Passport,

"5678",

"123456"

);

let car2 = new CarClass(

"Mercedes-Benz",

"E-Class",

2019,

"WDDZF8KBXKA123456",

"E234FG",

owner3,

CarBodyType.Sedan,

CarClassType.Business

);

let motorbike2 = new MotorbikeClass(

"Honda",

"CBR600RR",

2021,

"JH2PC3704MK012345",

"F987JK",

owner3,

"Алюминиевая рама",

true

);

let carStorage = new VehicleStorageClass<Car>();

carStorage.addVehicle(car2);

let motorbikeStorage = new VehicleStorageClass<Motorbike>();

motorbikeStorage.addVehicle(motorbike2);

console.log("Автомобили в хранилище:");

carStorage.getVehicles().forEach(vehicle => vehicle.printVehicleInfo());

console.log("Мотоциклы в хранилище:");

motorbikeStorage.getVehicles().forEach(vehicle => vehicle.printVehicleInfo());

// Задание 4

namespace Transport {

export enum myDocType {

Passport = "Паспорт",

DriverLicense = "Водительское удостоверение"

}

export interface Owner {

lastName: string;

firstName: string;

middleName: string;

birthDate: Date;

docType: myDocType;

docSeries: string;

docNumber: string;

printInfo(): void;

}

export interface Vehicle {

brand: string;

model: string;

year: number;

vin: string;

regNumber: string;

owner: Owner;

printVehicleInfo(): void;

}

export class OwnerClass implements Owner {

private \_lastName: string;

private \_firstName: string;

private \_middleName: string;

private \_birthDate: Date;

private \_docType: myDocType;

private \_docSeries: string;

private \_docNumber: string;

get lastName() {

return this.\_lastName;

}

get firstName() {

return this.\_firstName;

}

get middleName() {

return this.\_middleName;

}

get birthDate() {

return this.\_birthDate;

}

get docType() {

return this.\_docType;

}

get docSeries() {

return this.\_docSeries;

}

get docNumber() {

return this.\_docNumber;

}

set lastName(lastName: string) {

this.\_lastName = lastName;

}

set firstName(firstName: string) {

this.\_firstName = firstName;

}

set middleName(middleName: string) {

this.\_middleName = middleName;

}

set birthDate(birthDate: Date) {

this.\_birthDate = birthDate;

}

set docType(docType: myDocType) {

this.\_docType = docType;

}

set docSeries(docSeries: string) {

this.\_docSeries = docSeries;

}

set docNumber(docNumber: string) {

this.\_docNumber = docNumber;

}

constructor(

lastName: string,

firstName: string,

middleName: string,

birthDate: Date,

docType: myDocType,

docSeries: string,

docNumber: string

) {

this.\_lastName = lastName;

this.\_firstName = firstName;

this.\_middleName = middleName;

this.\_birthDate = birthDate;

this.\_docType = docType;

this.\_docSeries = docSeries;

this.\_docNumber = docNumber;

}

printInfo(): void {

console.log(`Владелец: ${this.\_lastName} ${this.\_firstName} ${this.\_middleName}`);

console.log(`Дата рождения: ${this.\_birthDate.toDateString()}`);

console.log(`Тип документа: ${this.\_docType}`);

console.log(`Серия документа: ${this.\_docSeries}`);

console.log(`Номер документа: ${this.\_docNumber}`);

}

}

export class VehicleClass implements Vehicle {

private \_brand: string;

private \_model: string;

private \_year: number;

private \_vin: string;

private \_regNumber: string;

private \_owner: Owner;

get brand() {

return this.\_brand;

}

get model() {

return this.\_model;

}

get year() {

return this.\_year;

}

get vin() {

return this.\_vin;

}

get regNumber() {

return this.\_regNumber;

}

get owner() {

return this.\_owner;

}

set brand(brand: string) {

this.\_brand = brand;

}

set model(model:string) {

this.\_model = model;

}

set year(year: number) {

this.\_year = year;

}

set vin(vin: string) {

this.\_vin = vin;

}

set regNumber(regNumber: string) {

this.\_regNumber = regNumber;

}

set owner(owner: Owner) {

this.\_owner = owner;

}

constructor(

brand: string,

model: string,

year: number,

vin: string,

regNumber: string,

owner: Owner

) {

this.\_brand = brand;

this.\_model = model;

this.\_year = year;

this.\_vin = vin;

this.\_regNumber = regNumber;

this.\_owner = owner;

}

printVehicleInfo(): void {

console.log(`Транспортное средство: ${this.\_brand} ${this.\_model}`);

console.log(`Год выпуска: ${this.\_year}`);

console.log(`VIN: ${this.\_vin}`);

console.log(`Регистрационный номер: ${this.\_regNumber}`);

}

}

export enum CarBodyType {

Sedan = "Седан",

Hatchback = "Хэтчбек",

Crossover = "Кроссовер",

}

export enum CarClassType {

Economy = "Эконом",

Business = "Бизнес",

Premium = "Премиум",

}

export interface Car extends Vehicle {

carBodyType: CarBodyType;

carClassType: CarClassType;

}

export class CarClass extends VehicleClass implements Car {

private \_carBodyType: CarBodyType;

private \_carClassType: CarClassType;

constructor(

brand: string,

model: string,

year: number,

vin: string,

regNumber: string,

owner: Owner,

carBodyType: CarBodyType,

carClassType: CarClassType

) {

super(brand, model, year, vin, regNumber, owner);

this.\_carBodyType = carBodyType;

this.\_carClassType = carClassType;

}

get carBodyType() {

return this.\_carBodyType;

}

get carClassType() {

return this.\_carClassType;

}

// Cеттеры

set carBodyType(carBodyType: CarBodyType) {

this.\_carBodyType = carBodyType;

}

set carClassType(carClassType: CarClassType) {

this.\_carClassType = carClassType;

}

printVehicleInfo(): void {

super.printVehicleInfo();

console.log(`Тип кузова: ${this.\_carBodyType}`);

console.log(`Класс автомобиля: ${this.\_carClassType}`);

}

}

export interface Motorbike extends Vehicle {

frameType: string;

isForSport: boolean;

}

export class MotorbikeClass extends VehicleClass implements Motorbike {

private \_frameType: string;

private \_isForSport: boolean;

constructor(

brand: string,

model: string,

year: number,

vin: string,

regNumber: string,

owner: Owner,

frameType: string,

isForSport: boolean

) {

super(brand, model, year, vin, regNumber, owner);

this.\_frameType = frameType;

this.\_isForSport = isForSport;

}

get frameType() {

return this.\_frameType;

}

get isForSport() {

return this.\_isForSport;

}

set frameType(frameType: string) {

this.\_frameType = frameType;

}

set isForSport(isForSport: boolean) {

this.\_isForSport = isForSport;

}

printVehicleInfo(): void {

super.printVehicleInfo();

console.log(`Тип рамы: ${this.\_frameType}`);

console.log(`Для спорта: ${this.\_isForSport ? "Да" : "Нет"}`);

}

}

export interface VehicleStorage<T extends Vehicle> {

creationDate: Date;

vehicles: T[];

getVehicles(): T[];

}

export class VehicleStorageClass<T extends Vehicle> implements VehicleStorage<T>

{

creationDate: Date;

vehicles: T[];

constructor() {

this.creationDate = new Date();

this.vehicles = [];

}

addVehicle(vehicle: T): void {

this.vehicles.push(vehicle);

}

getVehicles(): T[] {

return this.vehicles;

}

}

}

let owner4 = new Transport.OwnerClass(

"Кузнецов",

"Алексей",

"Алексеевич",

new Date(1993, 3, 15),

Transport.myDocType.Passport,

"9876",

"543210"

);

let car3 = new Transport.CarClass(

"Audi",

"A6",

2022,

"WAUZZZ4G0DN123456",

"H567JK",

owner4,

Transport.CarBodyType.Sedan,

Transport.CarClassType.Business

);

let motorbike3 = new Transport.MotorbikeClass(

"Kawasaki",

"Ninja ZX-10R",

2023,

"JKBZXV000NA123456",

"K123LM",

owner4,

"Трековая рама",

true

);

let carStorage2 = new Transport.VehicleStorageClass<Transport.Car>();

carStorage2.addVehicle(car3);

let motorbikeStorage2 = new Transport.VehicleStorageClass<Transport.Motorbike>();

motorbikeStorage2.addVehicle(motorbike3);

owner4.printInfo()

console.log("Автомобили в хранилище:");

carStorage2.getVehicles().forEach(vehicle => vehicle.printVehicleInfo());

console.log("Мотоциклы в хранилище:");

motorbikeStorage2.getVehicles().forEach(vehicle => vehicle.printVehicleInfo());

// Задание 5

import {TransportforLab2} from './lab2transport.js';

let owner5 = new TransportforLab2.OwnerClass(

"Иванов",

"Иван",

"Иванович",

new Date(1985, 5, 23),

TransportforLab2.myDocType.DriverLicense,

"4567",

"765432"

);

let car4 = new TransportforLab2.CarClass(

"Toyota",

"Camry",

2021,

"JTNBE46K703023456",

"L123MN",

owner5,

TransportforLab2.CarBodyType.Sedan,

TransportforLab2.CarClassType.Business

);

let motorbike4 = new TransportforLab2.MotorbikeClass(

"Ducati",

"Panigale V4",

2023,

"ZDM14BPW5NB123456",

"M654OP",

owner5,

"Монокок",

true

);

let carStorage3 = new TransportforLab2.VehicleStorageClass<TransportforLab2.Car>();

carStorage3.addVehicle(car4);

let motorbikeStorage3 = new TransportforLab2.VehicleStorageClass<TransportforLab2.Motorbike>();

motorbikeStorage3.addVehicle(motorbike4);

owner5.printInfo()

console.log("Автомобили в хранилище:");

carStorage3.getVehicles().forEach(vehicle => vehicle.printVehicleInfo());

console.log("Мотоциклы в хранилище:");

motorbikeStorage3.getVehicles().forEach(vehicle => vehicle.printVehicleInfo());