**ЛАБОРАТОРНА РОБОТА №4**

**КЛАСИ, ОБ'ЄКТИ І МЕТОДИ**

**Мета роботи:** ознайомитись з призначенням та особливостями геттерів (get) і сеттерів (set); знати алгоритми застосування геттерів (get) та сеттери (set) з метою обміну інформацією між полями об’єкту та іншими об’єктами.

**Постановка задачі**

Розберемо призначення сеттерів і геттерів на простому прикладі, без викладення будь-яких складностей.

Створимо головний клас Car.

**public class** Car {

**Public static void**main(String[] args) {

}

}

Створимо ще один клас Java і назвемо його BMW

**publicclass**BMW {

}

Уявімо, що ми розробили автомобіль BMW, неважливо якої серії. Але ми автори творіння і повинні знати його характеристики. При продажу автомобіля ми повинні сказати користувачеві, чим автомобіль заправляти і в яких умовах його можна експлуатувати.  
Бензин зазвичай йде за номерами. Введемо змінну палива fuel, встановимо мінімальну температуру нижче якої автомобіль не можна експлуатувати, ввівши змінну minTemp, тобто для цього об'єкта ми ввели два поля - змінні. Для наочності цим змінним привласнимо значення нуль.

**public class** BMW {

**private int** fuel =0;

**private int** minTemp=0;

}

Після цього створимо об'єкт bmw7,

**public class** Car {

**public static void** main(String[] args) {

// создание объекта автомобиль 7 модели

     BMWbmw7 = **new**BMW();

bmw7.fuel=92;

bmw7.minTemp=-52;

}

}

заправимо 92 бензином і вкажемо мінімальну температуру за бортом -520С. Користувач намагається використовувати машину при -520С, що абсолютно протипоказано. Машина природно не заводиться.

Це приклад того, що створюваний нами об'єкт може експлуатуватися в заборонених умовах.

Щоб цього не було створюються геттери і сеттери. Спробуємо, це зробити, перейшовши в клас BMW.

Перше, доступ до змінних безпосередньо здійснювати неможливо, необхідно встановити модифікатор доступу **private**. Змінні з таким класом можна використовувати тільки в межах свого класу.

Оголошення змінної без модифікатора або з public небажано:

1. Порушується безпека об'єкта.

2. Будь-який користувач може безпосередньо змінити властивість об'єкта.

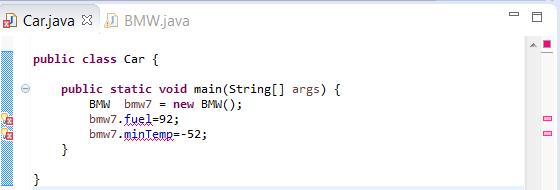
**public class** BMW {

**private int** fuel = 0;

**private int** minTemp= 0;

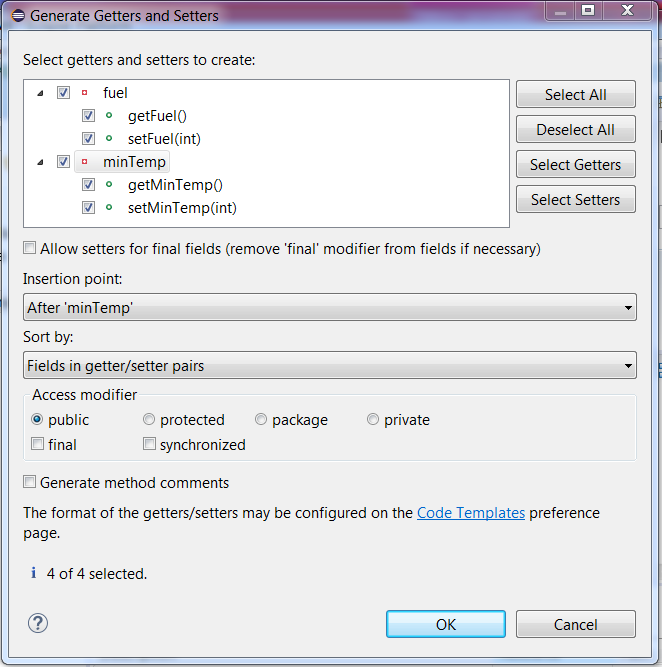
}

У головній програмі Car з'явилися помилки

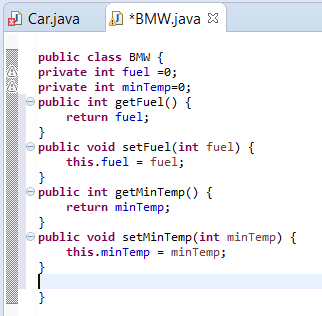


Далі ми чинимо так, генеруємо геттери і сеттери, використовуючи середовище розробки Eclipse. В пункті меню **Source** вибираємо команду **Generate Gettesand Setters...**

У вікні встановлюємо прапорці, так як показано на малюнку і натискаємо кнопку ОК.



В результаті виконання цієї програми в класі BMW будуть створені чотири нових методу c префіксами set - «встановити» і get - «взяти».



У классах***getFuel*** і ***getMinTemp*** значення змінних повертаються, в об'єкт,що викликає методи,через ім'я методу***get…*** з використанням ключового слова **return**.

Для класів ***setFuel*** і ***setMinTemp*** значенням змінних *fuel* і *minTemp* присвоюються значення переданих параметрів через наступну конструкцію

this.fuel = fuel

та

this.minTemp = minTemp

Коли користувач буде вводити номер палива, значення параметра буде перекривати значення змінної класу. Введене число буде перекривати 0.

Для контролю типу палива, що заправляється в метод ***setFuel***запровадинеобмеження на октанове число, а саме октанове число може перебувати в діапазоні [95; 98], що реалізується наступної програмної конструкцією

**if**(fuel<95 || fuel>98)

{System.***out***.println("Не тот вид палива");}

**else**

{System.***out***.println("Паливо у нормі");}

Також доопрацюємо метод ***setMinTemp***, обмеживши нижній температурний режим експлуатації автомобіля температурою -400С.

**if** (minTemp<40) {

System.out.println("Занадто холодно");

} **else** {

System.***out***.println("Температура в нормі. Можна їхати");}

Після всіх уточнень програмного коду класу BMW, його лістинг прийме наступний закінчений вигляд.

**public class** BMW {

**private int** fuel =0;

**private int** minTemp=0;

**public int** getFuel() {

**return**fuel;

}

**public void** setFuel(**int** fuel) {

**if**(fuel<95 || fuel>98)

{System.***out***.println("Нетот вид палива");}

**else**

{System.***out***.println("Паливоунормі");}

**this**.fuel = fuel;

}

**public int** getMinTemp() {

**Return** minTemp;

}

**public void** setMinTemp(**int** minTemp) {

**if** (minTemp<40) {

System.***out***.println("Занадто холодно");

} **else** {

System.***out***.println("Температура в нормі. Можна їхати");

}

**this**.minTemp = minTemp;

}

}

Тепер переходимо у наш головний клас **classCar**. Тут користувач не зможе безпосередньо звертатися до змінних, так як вони видні в межах свого класу. Тепер настав час прибрати старі звернення до змінних. У рядку, що звільнився ведемо ім'я об'єкту bmw7 і з клавіатури натиснемо клавішу з «точкою», відкриється додаткове вікно зі списком методів і ключових слів, пов'язаних із зазначеним об'єктом. Серед них будуть наші **set…** і **get…**.

Виберемо

bmw7.setFuel(92); // заливаємо 92 бензин

І будемо експлуатувати автомобіль при температурі -450Сbmw7.setMinTemp(-45);

Виводимо повідомлення на консоль використовуючи методи get.

System.out.println("-" + bmw7.getFuel ());

System.out.println("-" + bmw7.getMinTemp ());

Лістинг розробленої програми представлений нижче

**public class** Car {

**public static void** main(String[] args) {

BMW bmw7 = **new** BMW();

bmw7.setFuel(92);

System.***out***.println(" - " + bmw7.getFuel());

bmw7.setMinTemp(-45);

System.***out***.println(" - " + bmw7.getMinTemp());

}

}

**public class** BMW {

**private int** fuel =0;

**private int** minTemp=0;

**public int** getFuel() {

**return**fuel;

}

**public void** setFuel(**int**fuel) {

**if**(fuel<95 || fuel>98)

{System.***out***.print("Нетотвидпалива");}

**else**

{System.***out***.print("Паливовнормі");}

**this**.fuel = fuel;

}

**public int** getMinTemp() {

**retur**minTemp;

}

**public void** setMinTemp(**int** minTemp) {

**if** (minTemp<-40) {

System.***out***.print("Занадто холодно");

} **else** {

System.***out***.print("Температура в нормі. Можна їхати");

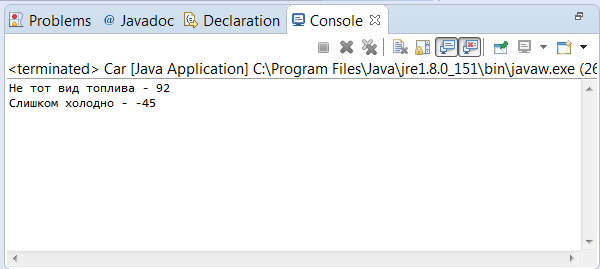
}

**this**.minTemp = minTemp;

}

}

Результат роботи програми:



**Висновки**

1. Ми створили код, коли користувач не може нашкодити програмі. Він не може безпосередньо звернутися до змінних, а тільки через **set** і **get**.

2. Якщо всередині коду поміняти ім'я змінної, то для користувача нічого не зміниться. Програма буде продовжувати працювати.

**ПРИКЛАДВИКОНАННЯ ІНДИВІДУАЛЬНИХ ЗАДАНЬ**

**Завдання**. Створити клас характеристик легкових машин, представлених для продажу, який складається з наступних полів: модель, колір, коробка передачі і тип палива. Необхідно згенерувати методи сеттерів і геттеров. Для сеттерів розробити конструктори, які реалізують обмеження, сформульовані для відповідних полів. Створити масив збережених об'єктів з підтримкою введення і виведення інформації на консоль.

**Короткий опис виконання роботи.**

Припустимо, що кожне із зазначених полів мають наступні обмеження:

модель (model) - BMW, VW, Skoda, Audio;

колір (color) - білий, чорний, вишневий, сірий;

коробка передач (transmis) - автомат, механіка;

паливо (fuel) - 92, 95, 98.

Створимо клас public class typeCar та поля з модифікаторами доступу **private**. Згенеруємо сеттери і геттери використовуючи середовище розробки Eclipse.

**publicclass** typeCar {

**private** String model="";// Марка автомобіля

**private** String color="";// Кольор автомобіля

**private** String transmis="";// Коробка передачь

**private int** fuel=0; // тип палива

**public** String getModel() {

**return** model;

}

**publicvoid** setModel(String model) {

**this**.model = model;

}

**public** String getColor() {

**return** color;

}

**publicvoid** setColor(String color) {

**this**.color = color;

}

**public** String getTransmis() {

**return** transmis;

}

**publicvoid** setTransmis(String transmis) {

**this**.transmis = transmis;

}

**publicint** getFuel() {

**return** fuel;

}

**publicvoid** setFuel(**int** fuel) {

**this**.fuel = fuel;

}

}

Методи **setModel( )**, **setFuel( )** і **setColor( )** доповнимо оператором **switch( )** для обробки обмежень.

Метод **setTransmis( )** використовує оператор **if( )** зі скороченим логічним оператором **&&.**

Для визначення події «порушення обмеження» одним з полів введена логічна змінна **carflag**, що повертає значення ***true***, якщо всі обмеження виконані.

Для повернення типу палива, щозаправляється, і рядка повідомлення про порушення обмеження, введена строкова змінна ***fuel2*** в методі **getFuel( )**.

Після всіх уточнень програмного коду класу **public class typeCar**, його лістинг прийме наступний закінчений вигляд

// Зберегаємий об'єкт

**public class** typeCar {

**private** String model=""; // марка автомобіля

**private** String color=""; // кольор автомобіля

**private** String transmis=""; // коробка передачь

**private int** fuel=0; // тип палива

**private** String fuel2=""; // тип палива у форматі String

**privateboolean** carflag=**false**; // прапорець невідповідності

**public** String getModel() {

**return** model;

}

**public void** setModel(String model) {

**switch** (model) {

**case** "BMV":

**case** "VW":

**case** "Skoda":

**case** "Audio":

carflag=**true**;

**break**;

**default**:

carflag=**false**;

**break**;

}

**if** (!carflag) {

model=model + " Машинатакоготипувідсутня";

}

**this**.model = model;

}

**public** String getColor() {

**return** color;

}

**public void** setColor(String color) {

**switch** (color) {

**case** "білий":

**case** "чорний":

**case** "вишневий":

**case** "сірий":

carflag=**true**;

**break**;

**default**:

carflag=**false**;

**break**;

}

**if** (!carflag) {

color=color + " Потрібнийкольорвідсутній";

}

**this**.color = color;

}

**public** String getTransmis() {

**return** transmis;

}

**public void** setTransmis(String transmis) {

**if**(transmis!="автомат" && transmis!="механіка"){

transmis=transmis + " Типкоробкипередачізаданоневірно" ;

carflag=**false**;}

**else**{

carflag=**true**;

}

**this**.transmis = transmis;

}

**public** String getFuel() {

**return** fuel2;

}

**public void** setFuel(**int** fuel) {

**switch** (fuel) {

**case** 92:

**case** 95:

**case** 98:

carflag=**true**;

**break**;

**default**:

carflag=**false**;

**break**;

}

fuel2= fuel + " ";

**if** (!carflag) {

fuel2=fuel + " Нетоттиппального";

}

**this**.fuel = fuel;

}

}

Створимо головний програму **Cars**. Тут користувач не зможе безпосередньо звертатися до полів об'єкта, так як їх видно тільки в межах свого класу.

У класі **Cars** створюється об'єктна змінна ***Avto***. У наступних рядках вводиться ім'я об'єкта **Avto**, потім з клавіатури вводиться точка. Відкривається додаткове вікно зі списком методів і ключових слів, пов'язаних із зазначеним об'єктом. Серед них будуть наші **set…** і **get…**. Лістинг **class Cars** представлено нижче

**publicclass** Cars {

**public static void** main(String[] args) {

// об'єктна змінна, яка описує параметри авто

typeCar Avto = **new** typeCar();

Avto.setModel("Audio");

System.***out***.println("Модель - " + Avto.getModel());

Avto.setColor("сірий");

System.***out***.println("Кольор - " + Avto.getColor());

Avto.setTransmis("автомат");

System.***out***.println("Коробкапередач - " + Avto.getTransmis());

Avto.setFuel(76);

System.***out***.println("Типпального - " + Avto.getFuel());

}

}

Далі створимо клас **public class** arrayCar масиву об'єктів, та введемо змінну arrCar з модифікатором доступу **private**

**private typeCar[ ] arrCar;**

і змінну лічильника

**private int nElems;**

яка буде потрібно для зручності роботи з масивом.

У конструкторі класу створюється масив об'єктів із заданим числом елементів

// Конструктор класу arrayCar

**public** arrayCar(**int** max){

arrCar=**new** typeCar[max]; // створеннямасивуоб'єктів

nElems=0; // лічильникобнуляється

}

Далі створюється метод для додавання об'єктів і назвемо його просто **Insert( )** з методом доступом **public** і ключовим словом **void** - метод нічого не повертатиме

**public void Insert (String m, String col, String kor, int top)**

В аргументах методу передаються вище наведені параметри легкових автомобілів.

**public void** Insert(String m, String col, String kor, **int** top){

**if**(nElems<arrCar.length){

arrCar[nElems] = **new** typeCar();

arrCar[nElems].setModel(m);

arrCar[nElems].setColor(col);

arrCar[nElems].setTransmis(kor);

arrCar[nElems].setFuel(top);

}

**else**

System.***out***.println("Масив повністю заповнено");

nElems++; // лічильник збільшується на одиницю

}

У тілі методу для початку потрібно створити логічну умову, яка перевіряє можливість додавання нових елементів об'єктів.

**if (nElems <arrCar.length) {**

Для кожного нового елемента-об'єкта створюється об'єктна змінна

**arrCar[nElems] = new typeCar( );**

і тільки після цього вводяться відповідні значення для полів об'єкта.

В останньому рядку методу змінна лічильника збільшується на одиницю

**nElems++;**

Для відображення полів, для всієї послідовності об'єктів, використовується метод **Display( )**.

**Public void** Display(){

**for** (**int** i=0; i<nElems;i++) {

System.***out***.println("Модель - " + arrCar[i].getModel());

System.***out***.println("Кольор - " + arrCar[i].getColor());

System.***out***.println("Коробка передач - "+arrCar[i].getTransmis());

System.***out***.println("Вид пального - "+ arrCar[i].getFuel());

System.***out***.println(arrCar[i].toString()+" "+ arrCar[i].hashCode());

}

}

У головному класі **public class Cars** створюється об'єкт класу **arrayCar** з передачею розмірності масиву. Далі з класу **arrayCar** викликається метод **arCars.Insert ("BMV", "білий", "автомат", 72);** зі значеннями відповідних параметрів легкового автомобіля.

**public static void** main(String[] args) {

arrayCar arCars;

arCars = **new** arrayCar(5);

arCars.Insert("BMV", "білий", "автомат", 72);

arCars.Insert("Audio", "чорний2", "механіка", 95);

arCars.Insert("VW", "вишневий", "автомат", 98);

arCars.Insert("Skoda", "сірий", "автомат3", 99);

arCars.Display();

}

Після всіх уточнень програмний код реалізації завдання прийме наступний закінченого вигляду

// Зберегаємий об'єкт

**publicclass** typeCar {

**private** String model="";// Марка автомобіля

**private** String color="";// Кольор автомобіля

**private** String transmis="";// Коробка апередач

**private int** fuel=0;// типпалива

**private** String fuel2=""; // тип палива у форматі String

**private boolean** carflag=**false**;// прапорець невідповідності

**public** String getModel() {

**return** model;

}

**Public void** setModel(String model) {

**switch** (model) {

**case** "BMV":

**case** "VW":

**case** "Skoda":

**case** "Audio":

carflag=**true**;

**break**;

**default**:

carflag=**false**;

**break**;

}

**if** (!carflag) {

model=model + " Машина такого типу відсутня";

}

**this**.model = model;

}

**public** String getColor() {

**return** color;

}

**Public void** setColor(String color) {

**switch** (color) {

**case** "білий":

**case** "чорний":

**case** "вишневий":

**case** "сірий":

carflag=**true**;

**break**;

**default**:

carflag=**false**;

**break**;

}

**if** (!carflag) {

color=color + " Потрібний кольор відсутній";

}

**this**.color = color;

}

**public** String getTransmis() {

**return** transmis;

}

**public void** setTransmis(String transmis) {

**if**(transmis!="автомат" && transmis!="механіка"){

transmis=transmis + " Тип коробки передачі задано невірно" ;

carflag=**false**;}

**else**{

carflag=**true**;

}

**this**.transmis = transmis;

}

**public** String getFuel() {

**return** fuel2;

}

**public void** setFuel(**int** fuel) {

**switch** (fuel) {

**case** 92:

**case** 95:

**case** 98:

carflag=**true**;

**break**;

**default**:

carflag=**false**;

**break**;

}

fuel2= fuel + " ";

**if** (!carflag) {

fuel2=fuel + " Не тот тип пального";

}

**this**.fuel = fuel;

}

}

**public class** arrayCar {

**private** typeCar[] arrCar;// змінна массиву, що зберегає

// об'єкти

**privateint** nElems; // лічильник елементів масиву

// Конструктор класу arrayCar

**public** arrayCar(**int** max){

arrCar=**new** typeCar[max]; // створення масиву об'єктів

nElems=0; // лічильник обнуляється

}

// МЕТОД

//заповнення полів поточного об'єкта масива

**public void** Insert(String m, String col, String kor, **int** top){

**if**(nElems<arrCar.length){

arrCar[nElems] = **new** typeCar();

arrCar[nElems].setModel(m);

arrCar[nElems].setColor(col);

arrCar[nElems].setTransmis(kor);

arrCar[nElems].setFuel(top);

}

**else**

System.***out***.println("Масив повністю заповнено");

nElems++; // лічильник збільшується на одиницю

}

// МЕТОД - вивід, по об'єктно, всіх полів на консоль

**public void** Display(){

**for** (**int** i=0; i<nElems;i++) {

System.***out***.println("Модель - "+arrCar[i].getModel());

System.***out***.println("Кольор - "+arrCar[i].getColor());

     System.***out***.println("Коробка передач - "+arrCar[i].getTransmis());

System.***out***.println("Вид пального - "+arrCar[i].getFuel());

    System.***out***.println(arrCar[i].toString()+" "+arrCar[i].hashCode());

}

}

}

**public class** Cars {

**public static void** main(String[] args) {

arrayCar arCars;

arCars = **new** arrayCar(5);

arCars.Insert("BMV", "білий", "автомат", 72);

arCars.Insert("Audio", "чорний2", "механіка", 95);

arCars.Insert("VW", "вишневий", "автомат", 98);

arCars.Insert("Skoda", "сірий", "автомат3", 99);

arCars.Display();

}

}

**ІНДИВІДУАЛЬНІ ЗАВДАННЯ**

Створити класи, специфікації яких наведені нижче. Визначити конструктори і методи **setТип(), getТип().** Задати обмеження, які можна накласти на поля класу. Визначити додатково методи в класі, що створює масив об'єктів.

|  |  |
| --- | --- |
| **№ варіанту** | **Зміст завдання** |
| 1 | Student: id, Прізвище, Ім'я, По батькові, Дата народження, Адреса, Телефон, Факультет, Курс, Група.  Створити масив об'єктів. Вивести: список студентів. |
| 2 | Customer: id, Прізвище, Ім'я, По батькові, Адреса, Номер кредитної картки, Номер банківського рахунку.  Створити масив об'єктів. Вивести: список покупців в алфавітному порядку. |
| 3 | Patient: id, Прізвище, Ім'я, По батькові, адресу, телефон, Номер медичної карти, Діагноз.  Створити масив об'єктів. Вивести список пацієнтів. |
| 4 | Abiturient: id, Прізвище, Ім'я, По батькові, адресу, телефон, Оцінки.  Створити масив об'єктів. Вивести список абітурієнтів в алфавітному порядку. |
| 5 | Book: id, Назва, Автор(и), Видавництво, Рік видання, Кількість сторінок, Ціна, Тип палітурки.  Створити масив об'єктів. Вивести список книг упорядкованих по авторам. |
| 6 | House: id, Номер квартири, Площа, Поверх, Кількість кімнат, Вулиця, Тип будівлі, Термін експлуатації. Створити масив об'єктів. Вивести: список квартир. |
| 7 | Phone: id, Прізвище, Ім'я, По батькові, Адреса, Номер кредитної картки, Дебет, Кредит, Час міських і міжміських розмов. Створити масив об'єктів. Вивести відомості про абонентів. |
| 8 | Car: id, Марка, Модель, Рік випуску, Колір, Ціна, Реєстраційний номер. Створити масив об'єктів. Вивести список автомобілів. |
| 9 | Product: id, Найменування, UPC, Виробник, Ціна, Термін зберігання, Кількість.  Створити масив об'єктів. Вивести список товарів для заданого найменування. |
| 10 | Train: Пункт призначення, Номер поїзда, Час відправлення, Число місць (загальних, купе, плацкарт, люкс). Створити масив об'єктів. Вивести список поїздів. |
| 11 | Bus: Прізвище та ініціали водія, Номер автобуса, Номер маршруту, Марка, Рік початку експлуатації, Пробіг. Створити масив об'єктів. Вивести список автобусів. |
| 12 | Airline: Пункт призначення, Номер рейсу, Тип літака, Час вильоту, Дні тижня.  Створити масив об'єктів. Вивести список рейсів. |
| 13 | Student: id, Прізвище, Ім'я, По батькові, Дата народження, Адреса, Телефон, Факультет, Курс, Група.  Створити масив об'єктів. Вивести: список студентів. |
| 14 | Customer: id, Прізвище, Ім'я, По батькові, Адреса, Номер кредитної картки, Номер банківського рахунку.  Створити масив об'єктів. Вивести: список покупців в алфавітному порядку. |
| 15 | Patient: id, Прізвище, Ім'я, По батькові, адресу, телефон, Номер медичної карти, Діагноз.  Створити масив об'єктів. Вивести список пацієнтів. |
| 16 | Abiturient: id, Прізвище, Ім'я, По батькові, адресу, телефон, Оцінки.  Створити масив об'єктів. Вивести список абітурієнтів в алфавітному порядку. |
| 17 | Book: id, Назва, Автор(и), Видавництво, Рік видання, Кількість сторінок, Ціна, Тип палітурки.  Створити масив об'єктів. Вивести список книг упорядкованих по авторам. |
| 18 | House: id, Номер квартири, Площа, Поверх, Кількість кімнат, Вулиця, Тип будівлі, Термін експлуатації. Створити масив об'єктів. Вивести: список квартир. |
| 19 | Phone: id, Прізвище, Ім'я, По батькові, Адреса, Номер кредитної картки, Дебет, Кредит, Час міських і міжміських розмов. Створити масив об'єктів. Вивести відомості про абонентів. |
| 20 | Car: id, Марка, Модель, Рік випуску, Колір, Ціна, Реєстраційний номер. Створити масив об'єктів. Вивести список автомобілів. |
| 21 | Product: id, Найменування, UPC, Виробник, Ціна, Термін зберігання, Кількість.  Створити масив об'єктів. Вивести список товарів для заданого найменування. |
| 22 | Train: Пункт призначення, Номер поїзда, Час відправлення, Число місць (загальних, купе, плацкарт, люкс). Створити масив об'єктів. Вивести список поїздів. |
| 23 | Bus: Прізвище та ініціали водія, Номер автобуса, Номер маршруту, Марка, Рік початку експлуатації, Пробіг. Створити масив об'єктів. Вивести список автобусів. |
| 24 | Airline: Пункт призначення, Номер рейсу, Тип літака, Час вильоту, Дні тижня.  Створити масив об'єктів. Вивести список рейсів. |
| 25 | Student: id, Прізвище, Ім'я, По батькові, Дата народження, Адреса, Телефон, Факультет, Курс, Група.  Створити масив об'єктів. Вивести: список студентів. |
| 26 | Customer: id, Прізвище, Ім'я, По батькові, Адреса, Номер кредитної картки, Номер банківського рахунку.  Створити масив об'єктів. Вивести: список покупців в алфавітному порядку. |
| 27 | Patient: id, Прізвище, Ім'я, По батькові, адресу, телефон, Номер медичної карти, Діагноз.  Створити масив об'єктів. Вивести список пацієнтів. |
| 28 | Abiturient: id, Прізвище, Ім'я, По батькові, адресу, телефон, Оцінки.  Створити масив об'єктів. Вивести список абітурієнтів в алфавітному порядку. |
| 29 | Book: id, Назва, Автор(и), Видавництво, Рік видання, Кількість сторінок, Ціна, Тип палітурки.  Створити масив об'єктів. Вивести список книг упорядкованих по авторам. |
| 30 | House: id, Номер квартири, Площа, Поверх, Кількість кімнат, Вулиця, Тип будівлі, Термін експлуатації. Створити масив об'єктів. Вивести: список квартир. |
| 31 | Phone: id, Прізвище, Ім'я, По батькові, Адреса, Номер кредитної картки, Дебет, Кредит, Час міських і міжміських розмов. Створити масив об'єктів. Вивести відомості про абонентів. |
| 32 | Car: id, Марка, Модель, Рік випуску, Колір, Ціна, Реєстраційний номер. Створити масив об'єктів. Вивести список автомобілів. |
| 33 | Product: id, Найменування, UPC, Виробник, Ціна, Термін зберігання, Кількість.  Створити масив об'єктів. Вивести список товарів для заданого найменування. |
| 34 | Train: Пункт призначення, Номер поїзда, Час відправлення, Число місць (загальних, купе, плацкарт, люкс). Створити масив об'єктів. Вивести список поїздів. |
| 35 | Bus: Прізвище та ініціали водія, Номер автобуса, Номер маршруту, Марка, Рік початку експлуатації, Пробіг. Створити масив об'єктів. Вивести список автобусів. |