Лабораторна робота №2

Класи, об’єкти. Масиви. Модифікатори доступу

Завдання

1. Створити проєкт, що складається з двох класів: основного (Main) та класу для представлення об’єкта відповідно специфікації, що наведена нижче. Кожний клас повинен бути розміщений в окремому пакеті. У створеному класі визначити приватні поля для зберігання указаних даних, конструктори для створення об’єктів та відкриті методи `setValue()`, `getValue()`, `toString()` для доступу до полів об’єкта.
2. В основному класі програми визначити методи, що створюють масив об’єктів. Задати критерії вибору даних та вивести ці дані на консоль. Для кожного критерію створити окремий метод.
3. Виконати програму, та пересвідчитись, що дані зберігаються та коректно виводяться на екран відповідно до вказаних критеріїв.
4. Розділити основний клас на декілька класів: клас-фабрику, що має статичний метод для створення об’єктів; клас-сервіс, що виконує запити; клас-контролер, який взаємодіє з користувачем; клас-вид, що візуалізує результати запитів.
5. Виконати змінену програму та пересвідчитись, що вона працює коректно.

Варіанти завдань

Варіант 1

Student: id, Повне Ім’я, Дата народження, Факультет, Група

Вивести:

1. Список студентів заданого факультету;
2. Список студентів, які народились після заданого року;
3. Список навчальної групи.

Варіант 2

Customer: id, Повне Ім’я, Номер кредитної картки, Баланс рахунку (кількість грошей)

Вивести:

1. Список покупців, чиє ім’я починається з указаних літер;
2. Список покупців, у яких номер кредитної картки знаходиться в заданому інтервалі;
3. Список покупців, які мають заборгованість (від’ємний баланс на карті)

Варіант 3

Patient: id, Повне Ім’я, Адреса, Телефон, Номер медичної карти, Діагноз.

Вивести:

1. Список пацієнтів, які мають вказаний діагноз;
2. Список пацієнтів, номер медичної карти у яких знаходиться в заданому інтервалі;
3. Кількість та список пацієнтів, номер телефона яких починається з вказаної цифри

Варіант 4

Abiturient: id, Повне Ім’я, Адреса, Телефон, Середній бал.

Вивести:

1. Список абітурієнтів із вказаним іменем;
2. Список абітурієнтів, середній бал у яких вище заданого;
3. Задане число n абітурієнтів, що мають найвищий середній бал.

Варіант 5

Book: id, Назва, Автор, Видавництво, Кількість сторінок.

Вивести:

1. Список книг заданого автора;
2. Список книг, що видані заданим видавництвом;
3. Список книг, що мають кількість сторінок не менше вказаної.

Варіант 6

House: id, Площа, Поверх, Кількість кімнат.

Вивести:

1. Список квартир, які мають задане число кімнат;
2. Список квартир, які розташовані на поверсі, що знаходиться в заданому проміжку;
3. Список квартир, які мають площу, що не менше заданої.

Варіант 7

Phone: id, Повне Ім’я, Номер рахунку, Час міських розмов, Час міжміських розмов.

Вивести:

1. Відомості про абонентів, у яких час міських розмов перевищує заданий;
2. Відомості про абонентів, які користувались міжміським зв’язком;
3. Відомості про абонентів чий номер рахунку знаходиться у вказаному діапазоні.

Варіант 8

Car: id, Модель, Рік випуску, Ціна, Реєстраційний номер.

Вивести:

1. Список автомобілів заданої моделі;
2. Список які експлуатуються більше n років;
3. Список автомобілів заданого року випуску, ціна яких більше вказаної.

Варіант 9

Product: id, Найменування, Ціна, Термін зберігання.

Вивести:

1. Список товарів для заданого найменування;
2. Список товарів для заданого найменування, ціна яких не перевищує задану;
3. Список товарів, термін зберігання яких більше заданого.

Варіант 10

Train: id, Пункт призначення, Номер поїзда, Час відправлення, Число місць.

Вивести:

1. Список поїздів, які прямують до заданого пункту призначення;
2. Список поїздів, які відправляються після заданої години;
3. Список поїздів, які прямують до заданого пункту призначення та кількість місць у яких не менше n.

Класи в Java

Класи Java можуть мати методи, і атрибути.

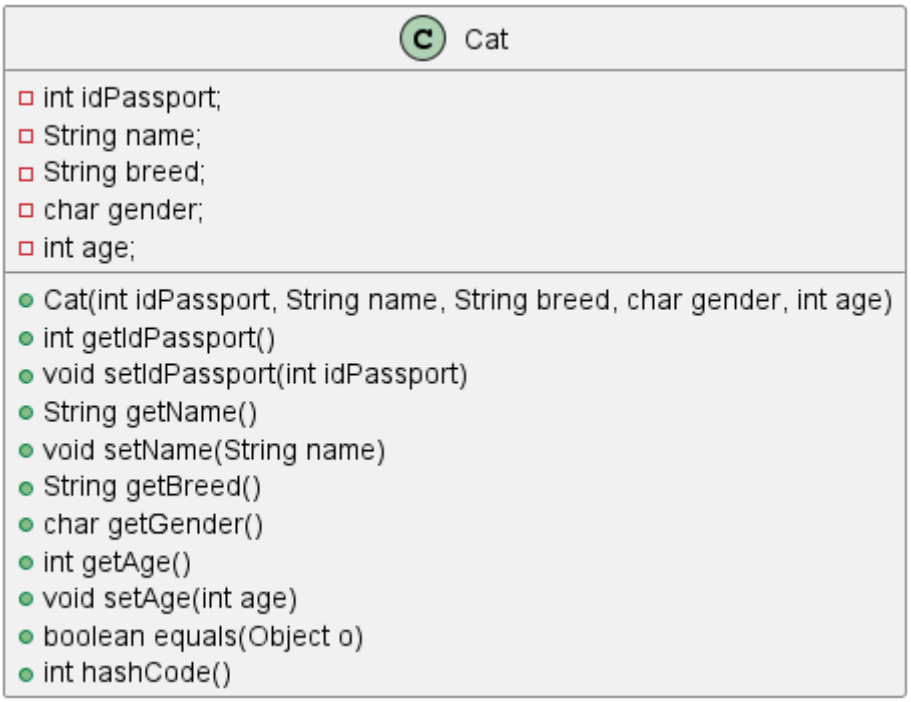
- Методи визначають поведінку об’єктів, що належать до цього класу.
- Змінні, або атрибути – це характеристики класу, що містять його дані

У Java прийнято, що атрибути класу, які можуть змінюватись, оголошуються з модифікатором доступу `private`, що не дозволяє їхнє використання «в обхід» спеціальних методів класу «геттерів» та «сеттерів». Своєю чергою, такі методи оголошуються з модифікатором `public`, що дозволить їхнє використання з методів інших класів.

Одним з методів класу, що використовується досить часто, є метод `equals()`. Цей метод дозволяє перевіряти об’єкти на рівність.

Слід зауважити, що проста перевірка об’єктів на рівність за допомогою операції == дає можливість перевірити лише той факт, що ми маємо справу з посиланнями на один і той самий об’єкт.

Приклад 1. ***Описання класу (у файлі Cat.java)***



Зверніть увагу, що сеттери визначені не для усіх полів!

```
package lab2.cats;

import java.util.Objects;

public class Cat {
    private int idPassport;
    private String name;
    private String breed;
    private char gender;
    private int age;

    public Cat(int idPassport, String name, String breed, char gender, int age) {
        this.idPassport = idPassport;
        this.name = name;
        this.breed = breed;
        this.gender = gender;
        this.age = age;
    }

    public int getIdPassport() {
        return idPassport;
    }

    public void setIdPassport(int idPassport) {
        this.idPassport = idPassport;
    }
```

```
    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getBreed() {
        return breed;
    }

    public char getGender() {
        return gender;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }

    @Override
    public String toString() {
        return "Cat{" +
                "idPassport=" + idPassport +
                ", name='" + name + '\'' +
                ", breed='" + breed + '\'' +
                ", gender=" + gender +
                ", age=" + age + '}';
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Cat cat = (Cat) o;
        return idPassport == cat.idPassport &&
                gender == cat.gender &&
                age == cat.age &&
                Objects.equals(name, cat.name) &&
                Objects.equals(breed, cat.breed);
    }
    @Override
    public int hashCode() {
        return Objects.hash(idPassport, name, breed, gender, age);
    }
}
```

Використання масивів

Типи масиву використовуються для визначення масивів – упорядкованих наборів однотипних змінних. Ви можете визначити масив над будь-яким наявним у мові типом, включаючи типи, визначені користувачем. Крім того, можна користатися масивами масивів чи багатовимірними масивами. Коротко говорячи, якщо ми можемо створити змінну деякого типу, виходить, ми можемо створити й масив змінних цього типу. Разом з тим створення масивів у мові Java може показатися вам незвичним, тому що воно вимагає застосування оператора `new`.

Приклад 2. ***Описання масивів і виділення пам'яті для масивів***

```
int[] myIntArray;        // описання масиву цілих чисел
myIntArray = new int[8]; // створення масиву з 8 цілих чисел
MyType[] myObjectArray;  // описання масиву об□єктів типу MyType
myObjectArray = new MyType[5]; // створення масиву з 5 елементів типу MyType
```

Оператор `new` дає команду оболонці часу виконання виділити необхідну кількість пам'яті під масив. Як видно з цього прикладу, не треба повідомляти розмір масиву тоді ж, коли ви створюєте змінну-масив. Після того, як ви створили масив оператором `new`, доступ до цього масиву здійснюється точно так само, як у мовах C чи Pascal.

Приклад 3. ***Присвоювання значень елементам масивів***

```
myIntArray[0] = 0;
myIntArray[1] = 1;
myIntArray[2] = 2;
myObjectArray[0] = new MyType();
myObjectArray[1] = new MyType();
myObjectArray[2] = new MyType();
myObjectArray[0].setValue(0);
myObjectArray[1].setValue(1);
myObjectArray[2].setValue(2);
```

Масиви в мові Java мають три важливих переваги перед масивами в інших мовах. По-перше, програмісту не треба вказувати розмір масиву при його оголошенні. По-друге, будь-який масив у мові Java є змінною - а це значить, що його можна передати як параметр методу і використовувати як значення, що повертається методом. І по-третє, завжди легко довідатися, який розмір даного масиву. Наприклад, так визначається розмір масиву, що був оголошений вище.

Приклад 4. ***Отримання довжини масиву***

```
int len = myIntArray.length;
System.out.println("Length of myIntArray=" + len);
```

Багатовимірні масиви у мові Java визначаються, як "масиви, елементами яких є масиви".

Тобто двовимірний масив – це масив, елементами якого є лінійні масиви. Наприклад, так відбувається робота з двовимірним масивом

Приклад 5. ***Описання та робота з двовимірним масивом***

```
double[][] m;
m = new double[3][4]; // масив з трьох рядків, у кожному по 4 елементи
m[1][3] = 5.4; // присвоювання значення елементу, що знаходиться у першому рядку під
номером 3
double[][] z;
z = new double[3][];  // массив з трьох рядків
z[0] = new double[1]; // у першому рядку  один елемент
z[1] = new double[2]; // у другому рядку  два
z[2] = new double[3]; // у третьому  три
z[2][2] = 1.5; // припустиме присвоювання
z[0][1] = 5.3; // ПОМИЛКА! У першому рядку є лише один елемент і з індексом 0
```

Приклад 6. ***Опис головного класу програми, що використовує клас*** `Cat.java` ***– у файлі*** `Main.java`

```
package lab2.demo;

import lab2.cats.Cat;

public class Main {

    public static void main(String[] args) {
       new Main().run();
    }

    private void run() {
        Cat[] cats = fillCatsArray();
        System.out.println("----------------------");
        printCats(cats);
        System.out.println("----------------------");
        printDvorCats(cats);
    }

    private void printDvorCats(Cat[] cats) {
        for (Cat cat : cats) {
            if (cat.getBreed().equals("Dvor")) {
                System.out.println(cat);
            }
        }
    }

    private void printCats(Cat[] cats) {
        for (Cat cat : cats) {
            System.out.println(cat);
```

```
        }
    }

    private Cat[] fillCatsArray() {
        return new Cat[]{
                new Cat(1, "Murka", "Sphinx", 'f', 1),
                new Cat(2, "Matroskin", "Dvor", 'm', 3),
                new Cat(3, "Felix", "Sibir", 'm', 2),
                new Cat(4, "Tom", "Dvor", 'm', 2)
        };
    }

}
```