Лабораторна робота №3

Класи, об'єкти. Масиви. Модифікатори доступу. Опрацювання рядків.

Тема: Розробка та реалізація програм із використанням класів для опису даних та створення масивів об'єктів.

Мета роботи: отримати навички створення та реалізації програм з використанням масивів об'єктів та вивчити методи роботи зі строковими даними.

Завдання.

1. Створити проект, що складається з двох класів: основного (Main) та класу для завдання згідно варіанту (таб.1).

2. Виконати програму, та пересвідчитись, що дані зберігаються та коректно виводяться на екран відповідно до вказаних критеріїв.

3. Створити проект2, що складається з двох класів: основного (Main) та класу для представлення об'єкта відповідно специфікації, що наведена нижче у таблиці 2. Кожний клас повинен бути розміщений у 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'єкту.

4. В основному класі програми визначити методи, що створюють масив об'єктів. Задати критерії вибору даних (див. Таб.2) та вивести ці дані на консоль. Для кожного критерію створити окремий метод.

5. Виконати програму, та пересвідчитись, що дані зберігаються та коректно виводяться на екран відповідно до вказаних критеріїв.

Варіанти завдань

Таблиця 1. Робота з рядками

В усіх варіантах завдань вважати, що текст складається з літер та пропусків. Словом будемо називати непорожню послідовність літер, яка не містить пропусків.

Додаткове завдання (для всіх варіантів)

Виконати теж саме завдання, вважаючи, що слова у текстах можуть містити лише літери, але відокремлюються не тільки пропусками, а й іншими символами, що не є літерами і цифрами (наприклад, знаки пунктуації).

Варіант 1. Ввести з клавіатури ціле число k та деяку літеру (символ). Після введення цих даних, ввести один рядок тексту. В кожному слові тексту k-у літеру замінити введеним. Якщо k більше довжини слова, коригування не виконувати. Вивести змінений текст.

Варіант 2. Ввести один рядок тексту. В тексті кожну літеру замінити її порядковим номером в алфавіті. Вивести вхідний рядок та отриманий рядок наступним чином: при виведенні в одному рядку друкувати текст з двома пропусками між літерами, в наступному рядку внизу під кожною літерою друкувати її номер.

Варіант 3. Ввести текст, що складається з декількох слів. В тексті після літери **Р** (або **р**), якщо вона не остання в слові, може зустрічатися літера **А** (або **а**). У такому випадку замінити її на літеру **О** (або **о**). Вивести початковий та змінений рядок.

Варіант 4. Ввести текст, що складається з декількох слів, ввести рядок, що буде використовуватися для заміни та довжину слів, які треба замінити. В тексті слова вказаної довжини замінити введеним рядком. Довжина рядка, який використовуватиметься для заміни, може не співпадати з довжиною слова. Вивести змінений рядок.

Варіант 5. Ввести текст, короткий рядок та ціле число k. В тексті після k-го символу вставити заданий короткий рядок. Вивести рядок після опрацювання та ціле число – кількість слів у отриманому рядку.

Варіант 6. Ввести текст і два коротких рядка s та t. Після кожного слова тексту, яке закінчується рядком s, вставити вказане слово t. Вивести початковий рядок та рядок після опрацювання.

Варіант 7. Ввести текст, один символ та ціле число k. В кожному слові тексту, що має довжину більшу k видалити введений символ. Вивести рядок до та після опрацювання.

Варіант 8. Ввести текст, що крім літер та пропусків може містити інші символи. Вилучити з нього всі символи, крім пропусків, які не є літерами. Між послідовностями літер, що йдуть безпосередньо одна за одною, залишити хоча би по одному пропуску. Вивести рядок до та після опрацювання.
Варіант 9. Ввести текст та ціле число – довжину слова. З тексту вилучити всі слова вказаної довжини, які починаються на приголосну літеру. Вивести початковий рядок та рядок після опрацювання.
Варіант 10. Ввести текст, що містить пару спеціальних символів, які вводяться (наприклад, дужки ‘(’ та ‘)’ або зірочки ‘*’ .). Видалити з тексту всі символи між вказаними символами. Вивести початковий рядок та рядок після опрацювання та число - кількість слів у опрацьованому рядку.

Таблиця 2. Специфікації та критерії вибору

№ вар.	***Завдання***
1.	**Customer**: id, Прізвище, Ім'я, По батькові, Адреса, Номер кредитної картки, Баланс рахунку (кількість грошей). Скласти масив об'єктів. Вивести: a) список покупців, із вказаним іменем; b) список покупців, у яких номер кредитної картки знаходиться в заданому інтервалі; c) кількість та список покупців, які мають заборгованість (від’ємний баланс на карті)
2.	**Student**: id, Прізвище, Ім'я, По батькові, Дата народження, Адреса, Телефон,Факультет, Курс, Група. Скласти масив об'єктів. Вивести: a) список студентів заданого факультету; b) список студентів, які народились після заданого року; c) список навчальної групи.

3.	**House**: id, Номер квартири, Площа, Поверх, Кількість кімнат, Вулиця. Скласти масив об'єктів. Вивести: a) список квартир, які мають задане число кімнат; b) список квартир, які мають задане число кімнат та розташовані на поверсі, який знаходиться в заданому проміжку; c) список квартир, які мають площу, що перевищує задану.
4.	**Car**: id, Модель, Рік випуску, Ціна, Реєстраційний номер. Скласти масив об'єктів. Вивести: a) список автомобілів заданої моделі; b) список автомобілів заданої моделі, які експлуатуються більше n років; c) список автомобілів заданого року випуску, ціна яких більше вказаної.
5.	**Book**: id, Назва, Автор, Видавництво, Рік видання, Кількість сторінок, Ціна. Скласти масив об'єктів. Вивести: a) список книг заданого автора; b) список книг, що видані заданим видавництвом; c) список книг, що випущені після заданого року.
6.	**Phone**: id, Прізвище, Ім'я, По батькові, Номер рахунку, Час міських розмов, Час міжміських розмов. Скласти масив об'єктів. Вивести: a) відомості про абонентів, у яких час міських розмов перевищує заданий; b) відомості про абонентів, які користувались міжміським зв'язком; c) відомості про абонентів чий номер рахунку знаходиться у вказаному діапазоні.
7.	**Abiturient**: id, Прізвище, Ім'я, По батькові, Адреса, Телефон, Середній бал. Скласти масив об'єктів. Вивести: a) список абітурієнтів із вказаним іменем; b) список абітурієнтів, середній бал у яких вище заданого; c) вибрати задане число n абітурієнтів, що мають найвищий середній бал.

8.	**Patient**: id, Прізвище, Ім'я, По батькові, Адреса, Телефон, Номер медичної карти, Діагноз. Скласти масив об'єктів. Вивести: a) список пацієнтів, які мають вказаний діагноз; b) список пацієнтів, номер медичної карти у яких знаходиться в заданому інтервалі; c) кількість та список пацієнтів, номер телефону яких починається з вказаної цифри
9.	**Product**: id, Найменування, Виробник, Ціна, Термін зберігання, Кількість. Скласти масив об'єктів. Вивести: a) список товарів для заданого найменування; b) список товарів для заданого найменування, ціна яких не перевищує задану; c) список товарів, термін зберігання яких більше заданого.
10.	**Train**: Пункт призначення, Номер поїзду, Час відправки, Число місць (загальних, купе, плацкарт, люкс). Скласти масив об'єктів. Вивести: a) список поїздів, які прямують до заданого пункту призначення; b) список поїздів, які прямують до заданого пункту призначення та відправляються після заданої години; c) список поїздів, які відправляються до заданого пункту призначення та мають загальні місця.

Теоретичні відомості

Рядок у мові Java - це основний носій текстової інформації. Це не масив символів типу char, а об'єкт відповідного класу. Системна бібліотека Java містить класи String, StringBuilder і StringBuffer, що підтримують роботу з рядками і визначені в пакеті java.lang, що підключається автоматично. Ці класи оголошені як final, що означає неможливість створення власних породжених класів з властивостями рядків.

Клас String

Кожен рядок, створюваний за допомогою оператора **new** або за допомогою літерала (укладений в подвійні апострофи), є об'єктом класу String. Особливістю об'єкта класу String є те, що його значення не може бути змінено після створення об'єкту. При виклику будь-якого методу класу, результатом може бути новий рядок, але не змінюється початковий. Таким чином будь-який метод, що намагається змінити рядок приводить до створення нового об'єкта.

Клас String підтримує кілька конструкторів, наприклад:

```
String(),

String(String str),

String(byte asciichar[]),

String(char[] unicodechar),

String(StringBuffer sbuf),

String(StringBuilder sbuild)
```
та інші.

Коли Java зустрічає літерал, укладений в подвійні лапки, автоматично створюється об'єкт типу String, на який можна встановити посилання. Таким чином, об'єкт класу String можна створити, присвоюючи посиланню на клас значення існуючого літерала, або за допомогою оператора **new** і конструктора, наприклад:

```
String s1 = "www.berkut.mk.ua";

String s2 = new String("www.berkut.mk.ua ");
```

Клас String містить наступні (тут наведені основні, але не всі) методи для роботи з рядками:

`String concat(String s)` або “+” – злиття рядків; **`boolean`**

`equals(Object ob)` та `equalsIgnoreCase(String s)` – порівняння

рядків з урахуванням та без урахування регістра відповідно;

`int` `compareTo(String s)` та `compareToIgnoreCase(String s)` – лексикографічне порівняння рядків з урахуванням та без урахування регістра.

Метод виконує віднімання кодів символів рядка, що викликає метод та рядка, що передається до методу та повертає ціле значення. Метод повертає значення нуль у випадку, якщо equals() повертає значення true; **boolean** `contentEquals(StringBuffer sb)` – порівняння рядка та вмісту об'єкта типу StringBuffer;

`String substring(`**int** `n,` **int** `m)` – отримання з рядку підрядка, починаючи з позиції n (включаючи) до позиції m (не включаючи). Нумерація символів в рядку починається з нуля;

`String substring(`**int** `n)` – отримання з рядку підрядка, починаючи з позиції n; **int** `length()` – визначення довжини рядка; **int** `indexOf(`**char** `ch)` – визначення позиції символу у рядку; **int** `indexOf(`**String** `s)` – визначення позиції підрядку у рядку; **int** `indexOf(`**String** `s,` **int** `fromIndex)` – визначення позиції підрядку у рядку починаючи з вказаного індексу; **int** `lastIndexOf(`**char** `ch)` – визначення останньої позиції символу у рядку; **int** `lastIndexOf(`**String** `s)` – визначення останньої позиції підрядку у рядку;

int `lastIndexOf(`**String** `s,` **int** `fromIndex)` – визначення останньої позиції підрядку у рядку але не більше вказаного індексу; **static** `String valueOf(значення)` – перетворення змінної примітивного типу

у рядок;

`String toUpperCase()/toLowerCase()` – перетворення всіх символів рядка, який викликає метод у верхній/нижній регістр;

`String replace(`**char** `c1,` **char** `c2)` – заміна у рядку всіх входжень першого символу другим символом;

`String replace(CharSequence target, CharSequence replacement)` – заміна у рядку всіх входжень першого символу другим символом `String trim()` – вилучення всіх пропусків на початку та в кінці

рядка; **char** charAt(**int** position) – отримання символу із вказаної позиції (нумерація з нуля);

boolean isEmpty() – повертає true, якщо довжина рядка дорівнює 0;

static String format(String format, Object... args),– генерує форматований рядок, отриманий з використанням формату.

String[] split(String regex) – пошук входження в рядок заданого регулярного виразу (розділителя або розділителів) та поділ початкового рядка у відповідності з цим на масив рядків.

String[] split(String regex, **int** limit) – як і попередній метод, але кількість елементів у масиві не може перевищувати limit **boolean** startsWith(String prefix) – перевірка, чи починається рядок із вказаного префіксу

boolean endsWith(String suffix) – перевірка, чи закінчується рядок на вказаний суфікс

Класи StringBuilder та StringBuffer

Класи StringBuilder та StringBuffer є “близнюками” та за призначенням наближені до класу String, але, на відмінність від останнього, вміст та розміри об'єктів класів StringBuilder та StringBuffer можна змінювати. Тому, якщо потрібно виконувати багато операцій з перетворення рядків, треба в першу чергу, розглядати можливість використання саме об’єктів класів StringBuilder та StringBuffer.

За допомогою відповідних методів та конструкторів об'єкти класів StringBuffer,

StringBuilder та String можна перетворювати один до одного. Конструктор класу StringBuffer (так саме як і StringBuilder) може приймати в якості параметра об'єкт String або невід'ємний розмір буфера. Об'єкти цього класу можна перетворювати в об'єкт класу String методом toString() або за допомогою конструктора класу String.

Слід звернути увагу на такі методи: **void** setLength(**int** n) – установка розміру буфера; **void** ensureCapacity(**int** minimum) – установка гарантованого мінімального розміру буфера; **int** capacity() – отримання поточного розміру буфера;

StringBuilder append(параметри) – додавання до вмісту об'єкта рядкового представлення аргументу, який може бути символом, значенням примітивного типу, масивом та рядком;

StringBuilder insert(параметри) – вставка символу, об'єкта або рядка в указану позицію;

StringBuilder deleteCharAt(**int** index) – вилучення символу;

StringBuilder delete(**int** start, **int** end) – вилучення підрядку;

StringBuilder reverse() – перестановка вмісту об'єкта у зворотному порядку.

void setCharAt(int index, char ch) – заміна символа у вказаній позиції.

В класах StringBuilder та StringBuffer присутні також методи, аналогічні методам класу String, такі як replace(), substring(), charAt(), length(), getChars(), indexOf() та ін

Класи в Java

Класи Java можуть мати методи, і атрибути.

• Методи визначають, що клас може зробити.

• Атрибути – це характеристики класу

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». В свою чергу, 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 оди і той самий об’єкт.

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

```
package lab2;

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 void setBreed(String breed) {
        this.breed = breed;
    }

    public char getGender() {
        return gender;
    }

    public void setGender(char gender) {
        this.gender = 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 існуючим 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 і 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 3. 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 4. 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 5. Опис головного класу програми, що використовує клас Cat.java – у файлі Main.java

```
package main;

import lab2.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 (int i = 0; i < cats.length; i++) {
            if (cats[i].getBreed().equals("Dvor")) {
                System.out.println(cats[i]);
            }
        }
    }

    private void printCats(Cat[] cats) {
        for (int i = 0; i < cats.length; i++) {
            System.out.println(cats[i]);
        }
    }

    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)
        };
    }

}
```

Повністю проект цієї програми можна скачати за посиланням
http://www.berkut.mk.ua/download/files/java/lab2.zip