

### **Лабораторна робота 13.**

**Паралельне виконання. Багатопоточність. Ефективність використання.**  
Мета:

- Ознайомлення з моделлю потоків Java
- Організація паралельного виконання декількох частин програми.
- Вимірювання часу паралельних та послідовних обчислень.
- Демонстрація ефективності паралельної обробки.

Вимоги:

Використовуючи програми рішень попередніх задач, продемонструвати можливість паралельної обробки елементів контейнера: створити не менше трьох додаткових потоків, на яких викликати відповідні методи обробки контейнера.

Забезпечити можливість встановлення користувачем максимального часу виконання при закінченні якої обробка повинна припинятися незалежно від того знайдений кінцевий результат чи ні.

Для паралельної обробки використовувати алгоритми, що не змінюють початкову колекцію.

Кількість елементів контейнера повинна бути досить велика, складність алгоритмів обробки колекції повинна бути зіставна, а час виконання приблизно однаковий, наприклад:

- пошук мінімуму або максимуму;
- обчислення середнього значення або суми;
- підрахунок елементів, що задовольняють деякій умові;
- відбір за заданим критерієм;
- власний варіант, що відповідає обраній прикладної області.

Забезпечити вимірювання часу паралельної обробки елементів контейнера за допомогою розроблених раніше методів.

Додати до алгоритмів штучну затримку виконання для кожної ітерації циклів поелементної обробки контейнерів, щоб загальний час обробки був декілька секунд.

Реалізувати послідовну обробку контейнера за допомогою методів, що використовувались для паралельної обробки та забезпечити вимірювання часу їх роботи.

Порівняти час паралельної і послідовної обробки та зробити висновки про ефективність розпаралелювання:

- результати вимірювання часу звести в таблицю;
- обчислити та продемонструвати у скільки разів паралельне виконання швидше послідовного.

Виконала:

студентка групи КН-108 Бокшо Каріна Емеріхівна

## Прикладна задача

Бюро знайомств. Запис про клієнта: стать; реєстраційний номер; дата реєстрації; відомості про себе (довільний набір властивостей: ім'я, зріст, колір очей, дата народження, хобі тощо); вимоги до партнера (довільний набір властивостей).

## Опис програми

Програма реалізована у вигляді інтерактивного консольного вікна з діалоговим режимом роботи з користувачем. Основне призначення: демонстрація управління масивом domain-об'єктів. Реалізовано додавання та генерування нових об'єктів, видалення, показ інформації.

## Важливі фрагменти програми

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.LinkedList;
import java.util.Scanner;

import ua.lpnuai.oop.boksho.util.Console;

/**
 * Забезпечує обробку команд користувача у вигляді інтерактивного діалогового
 * меню.
 */
public class DialogHelper {
    /**
     * Перелік команд
```

```

*
* @author student Lytvyn I.I. KIT-26A
*/
public enum ACTION {
    add, generate, clean, count, exit, read, remove, save, search, sort,
show
}

/**
 *
 */
static int DEFAULT_SIZE = 10;

/**
 * Бюро знайомств
 */
static Bureau bureau = new Bureau();

/**
 * Клієнти
 */
static LinkedList<Client> clients = new LinkedList<>();

/**
 * Розмежувач
 */
public static String LINE = "-----"
    + "-----\n";

/**
 * Сканування вводу
 */
public static Scanner sc = new Scanner(System.in);

/**
 * Підтвердження завершення роботи
 */

```

```

public static boolean exit = false;

/**
 * Реєстраційний номер
 */
private static int regNum = 0;

/**
 * Додавання нового клієнту
 */
public static void add() {
    final Client temp = ClientGenerator.build();
    if (temp != null) {
        temp.setRegNum(regNum);
        clients.addLast(temp);
        bureau.setClients(clients);
        regNum++;
    } else {
        System.out.println("Введено некоректні дані!");
    }
}

/**
 * Генерування нових клієнтів
 *
 * @param count
 *         кількість клієнтів
 */
public static void generate(int count) {
    clients = new LinkedList<>(ClientGenerator.buildClients(count));
    regNum = clients.size();
    bureau.setClients(clients);
}

```

```

/**
 * Отримує відповідь від користувача
 *
 * @return answer відповідь від користувача
 * @throws IOException
 *
 *          виняткова ситуація при роботі з вводом
 */
public static String getInput() throws IOException {
    System.out.format("\nВаша відповідь: ");
    final BufferedReader br = new BufferedReader(
        new InputStreamReader(System.in));
    final String answer = br.readLine(); // Запис тексту до буферу
    return answer;
}

```

```

/**
 * Виконує обробку введеної команди
 *
 * @param action
 *
 *          введена команда
 * @throws Exception
 *
 *          будь-яка виникаюча виняткова ситуація
 */
public static void handleAction(ACTION action) throws Exception {
    switch (action) {
        case add:
            add();
            break;
        case generate:
            System.out.println("Введіть кількість клієнтів.");
            final String size = getInput();
            final int count = Integer.parseInt(size);
            generate(count);
            break;
    }
}

```

```

case clean:
    clients.clear();
    regNum = clients.size();
    bureau.setClients(clients);
    break;
case count:
    count(10000);
    break;
case read:
    /* Зчитування екземпляру контейнеру */
    System.out.println("Виберіть варіант:");
    System.out.println("1. Стандартна серіалізація");
    System.out.println("2. Без використання протоколу серіалізації");
    final int serNum = Integer.parseInt(getInput());
    if (serNum == 1) {
        read(true, null);
    } else if (serNum == 2) {
        read(false, null);
    } else {
        throw new IllegalArgumentException();
    }
    break;
case remove:
    /* Видалення */
    System.out.println("Введіть реєстраційний номер");
    final String num = getInput();
    final int regNum = Integer.parseInt(num);
    final Client remove = ClientGenerator.build();
    remove.setRegNum(regNum);
    remove(remove);
    break;
case save:
    /* Запис контейнеру */
    System.out.println("Виберіть варіант:");

```

```

        System.out.println("1. Стандартна серіалізація");
        System.out.println("2. Без використання протоколу серіалізації");
        final int dserNum = Integer.parseInt(getInput());
        if (dserNum == 1) {
            save(true, null);
        } else if (dserNum == 2) {
            save(false, null);
        } else {
            throw new IllegalArgumentException();
        }
        break;
case search:
    search();
    break;
case sort:
    /* Запис контейнеру */
    System.out.println("Виберіть варіант:");
    System.out.println("1. date");
    System.out.println("2. hobby");
    System.out.println("3. reqs");
    final int sortNum = Integer.parseInt(getInput());
    if (sortNum == 1) {
        ClientUtil.sort(clients,
            ComparatorFactory.getComparator("date"));
    } else if (sortNum == 2) {
        ClientUtil.sort(clients,
            ComparatorFactory.getComparator("hobby"));
    } else if (sortNum == 3) {
        ClientUtil.sort(clients,
            ComparatorFactory.getComparator("reqs"));
    } else {
        throw new IllegalArgumentException();
    }
    bureau.setClients(clients);

```

```

        break;
    case show:
        show();
        break;
    case exit:
        exit = true;
        break;
    default:
        System.err.println("Error data ^-(");
        break;
    }
}

/**
 * @param isStandart
 *      чи стандартна серіалізація
 * @param path
 */
public static void read(boolean isStandart, String path) {
    if (isStandart) {
        clients = SerializationUtil.deserialize(path);
    } else {
        clients = SerializationUtil.read(path);
    }
    bureau.setClients(clients);
}

/**
 * Видалення клієнту
 *
 * @param client
 *      який потрібно видалити
 */
public static void remove(Client client) {

```



```

        clients.remove(client);
        bureau.setClients(clients);
    }

    /**
     * @param isStandart
     *      чи стандартна десеріалізація
     * @param path
     */
    public static void save(boolean isStandart, String path) {
        if (isStandart) {
            SerializationUtil.serialize(bureau.getClients(), path);
        } else {
            SerializationUtil.save(bureau.getClients(), path);
        }
    }

    public static void search() {
        if (bureau.getClients().isEmpty()) {
            System.out.println("Контейнер порожній!");
        } else {
            final LinkedList<Client> result = ClientUtil.search(
                bureau.getClients());
            if (result.isEmpty()) {
                System.out.println("Відповідностей не знайдено!");
            } else {
                Console.clean();
                /* Виведення клієнтів */
                System.out.format("\nЗнайдено пар: %d\n\n", result.size());
                for (int i = 0; i < result.size(); i++) {
                    System.out.println("Знайдена пара:");
                    System.out.println(ClientUtil.info(result.get(i++)));
                    System.out.println(ClientUtil.info(result.get(i)));
                }
            }
        }
    }

```

```

        SerializationUtil.save(result,
                                Explorer.defaultPath("save.ser"));
    }
}

```

```
/**
```

```
 * Показ клієнтів
```

```
 */
```

```

public static void show() {
    if (bureau.getClients().isEmpty()) {
        System.out.println("Контейнер порожній!");
    } else {
        /* Виведення клієнтів */
        System.out.println("\nПоточний вміст контейнеру:\n");
        System.out.println(bureau);
    }
}

```

```
/**
```

```
 * Реалізує інтерактивне діалогове меню для забезпечення отримання команд
```

```
 * від користувача
```

```
 *
```

```
 * @param mode
```

```
 * @param initSize
```

```
 */
```

```

public static void start(int mode, int initSize) {
    Console.clean();
    switch (mode) {
        case 0:
            System.out.println(
                "#-----#");
            System.out.println("| Використовується звичайний режим! |");
            System.out.println(

```

```

        "#-----#");
    break;
case 1:
    if (Explorer.auto()) {
        System.out.println(
            "#-----
#");

        System.out.println(
            "| Знайдено список клієнтів в автоматичному режимі!
|");

        System.out.println(
            "#-----
#");

        read(false, Explorer.DEFAULT_PATH);
        if (clients.size() == 0) {
            System.out.println("Дані відсутні! "
                + "Буде згенеровано список клієнтів!");
            generate(DEFAULT_SIZE);
        }
        save(false, Explorer.defaultPath("data.ser"));
    }
    break;
case 2:
    System.out.println(
        "#-----
#");

    System.out.println(
        "| Згенеровано список клієнтів в автоматичному режимі!
|");

    System.out.println(
        "#-----
#");

    if (initSize == 0) {
        generate(DEFAULT_SIZE);
    } else {
        generate(initSize);
    }
}

```

```

        save(false, Explorer.defaultPath("data.ser"));
        break;
    default:
        break;
    }
    Console.pause();
    do {
        Console.clean();
        comands();
        System.out.format("\nВведіть команду: ");
        sc = new Scanner(System.in);
        final ACTION cur = ACTION.valueOf(sc.next());
        try {
            handleAction(cur);
            Console.pause();
        } catch (final Exception ex) {
            System.out.println(ex.toString());
        }
    } while (!exit);
}

/**
 * Список команд очищення контейнера, перетворення у масив, перетворення у
 * рядок, перевірку на наявність елементів
 */
private static void comands() {
    System.out.format("Список доступних команд:\n\n");
    System.out.format("add - додавання нового клієнту\n");
    System.out.format("generate - додавання згенерованих клієнтів\n");
    System.out.format("read - зчитування бюро з файлу\n");
    System.out.format("remove - видалення клієнту\n");
    System.out.format("show - перегляд бюро\n");
    System.out.format("sort - сортування бюро\n");
    System.out.format("search - пошук клієнтів\n");

```

```

        System.out.format("save - збереження бюро\n");
        System.out.format("exit - завершення програми\n");
        System.out.format("clean - очищення клієнтів\n");
        System.out.format("count - мультипоточкова обробка\n");
        System.out.format("sort - сортування клієнтів\n");
    }

    /**
     *
    */
    private static void count(int timeout) {
        final Thread first = new Thread(new Runnable() {
            @Override
            public void run() {
                CountTask.countMinAge(clients);
            }
        });

        final Thread second = new Thread(new Runnable() {
            @Override
            public void run() {
                CountTask.countMaxAge(clients);
            }
        });

        final Thread third = new Thread(new Runnable() {
            @Override
            public void run() {
                CountTask.countAvrAge(clients);
            }
        });

        try {
            first.join(timeout);

```

```

        first.start();

        second.join(timeout);
        second.start();

        third.join(timeout);
        third.start();
    } catch (final InterruptedException e) {
        e.printStackTrace();
    }
}
}

```

### 3. РЕЗУЛЬТАТ РОБОТИ

```

C:\WINDOWS\system32\cmd.exe
Список доступних команд:
add - додавання нового клієнта
generate - додавання згенерованих клієнтів
remove - видалення клієнта
show - перегляд клієнтів
exit - завершення програми

Введіть команду: add

Введіть стать.
Ваша відповідь: чоловік

Введіть ім'я.
Ваша відповідь: Денис

Введіть зріст.
Ваша відповідь: 188

Введіть колір очей.
Ваша відповідь: блакитний

Введіть дату народження у форматі dd.MM.yyyy.
Ваша відповідь: 08.05.1999

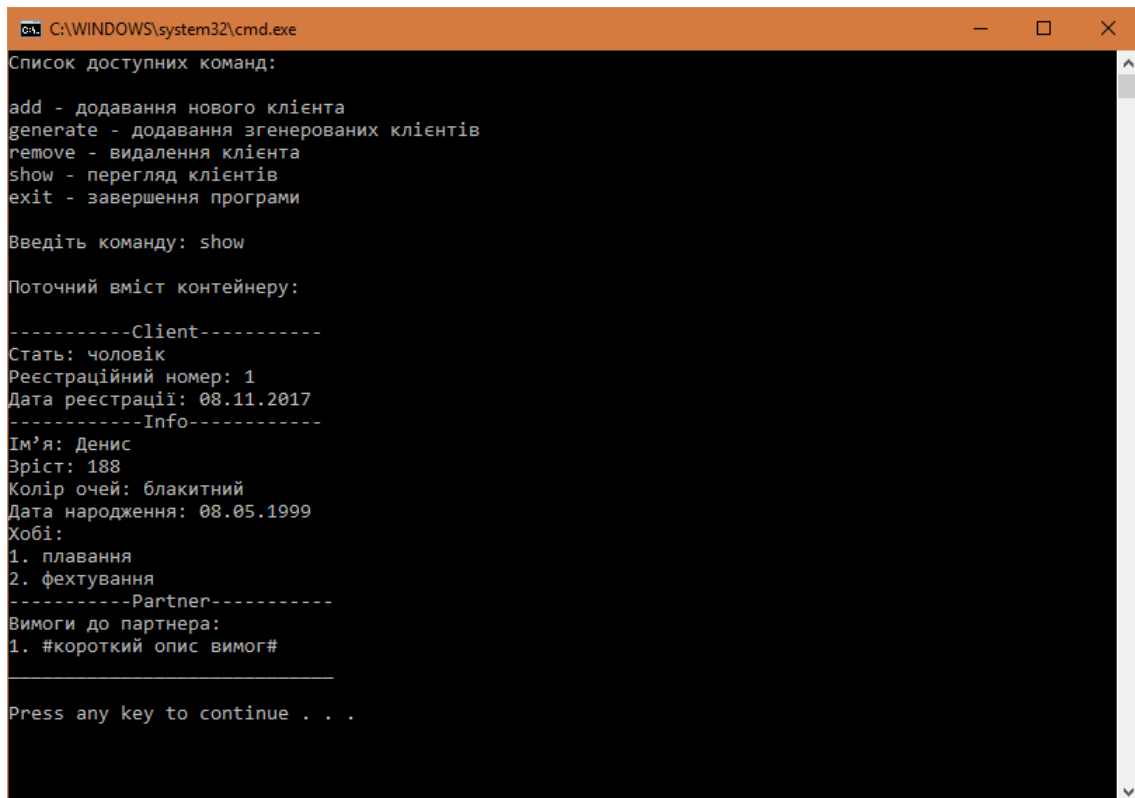
Введіть хобі через ";".
Ваша відповідь: плавання;фехтування

Введіть вимоги до партнера через ";".
Ваша відповідь: #короткий опис вимог#
Press any key to continue . . .

```

Рисунок 2 "Результати"

Для налагодження роботи програми було успішно проведено її тестування.



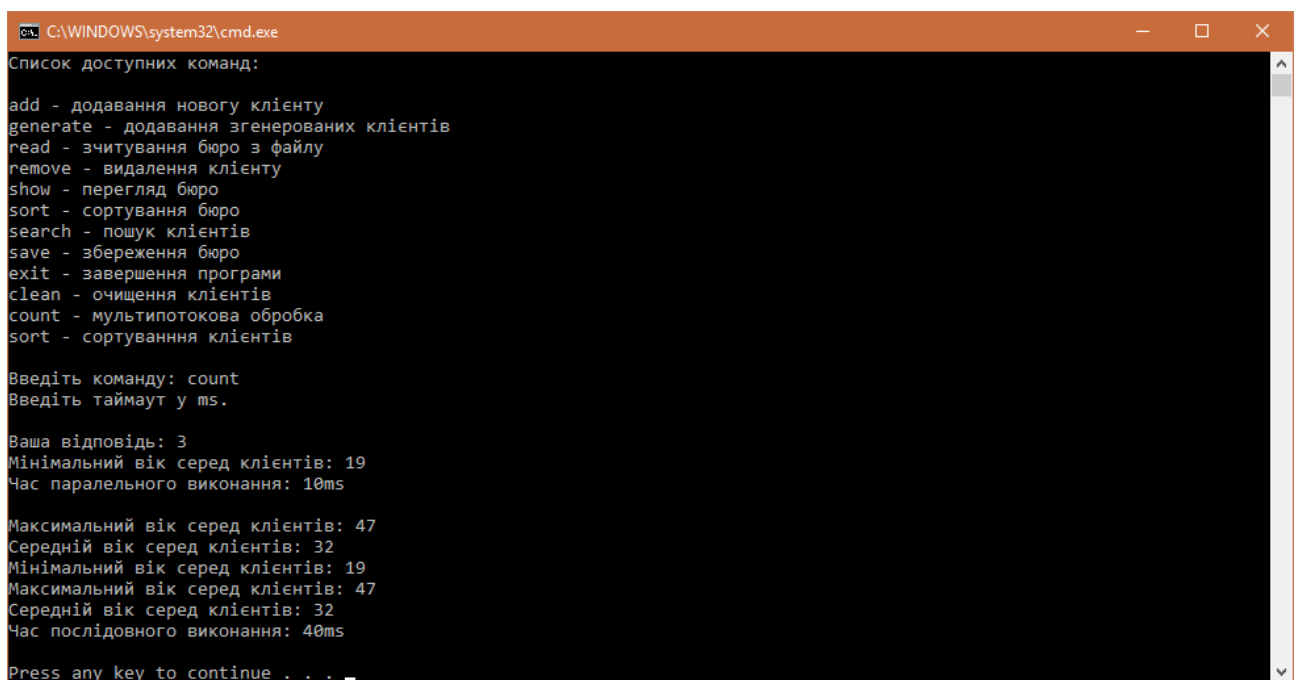
```
C:\WINDOWS\system32\cmd.exe
Список доступних команд:
add - додавання нового клієнта
generate - додавання згенерованих клієнтів
remove - видалення клієнта
show - перегляд клієнтів
exit - завершення програми

Введіть команду: show

Поточний вміст контейнеру:
-----Client-----
Стать: чоловік
Реєстраційний номер: 1
Дата реєстрації: 08.11.2017
-----Info-----
Ім'я: Денис
Вік: 188
Колір очей: блакитний
Дата народження: 08.05.1999
Хобі:
1. плавання
2. фехтування
-----Partner-----
Вимоги до партнера:
1. #короткий опис вимог#

Press any key to continue . . .
```

Рисунок 3 "Результати"



```
C:\WINDOWS\system32\cmd.exe
Список доступних команд:
add - додавання нового клієнту
generate - додавання згенерованих клієнтів
read - зчитування бюро з файлу
remove - видалення клієнту
show - перегляд бюро
sort - сортування бюро
search - пошук клієнтів
save - збереження бюро
exit - завершення програми
clean - очищення клієнтів
count - мультипоточкова обробка
sort - сортування клієнтів

Введіть команду: count
Введіть таймаут у ms.

Ваша відповідь: 3
Мінімальний вік серед клієнтів: 19
Час паралельного виконання: 10ms

Максимальний вік серед клієнтів: 47
Середній вік серед клієнтів: 32
Мінімальний вік серед клієнтів: 19
Максимальний вік серед клієнтів: 47
Середній вік серед клієнтів: 32
Час послідовного виконання: 40ms

Press any key to continue . . .
```

Рисунок 4 "Результати"

## ВИСНОВКИ

Створено і налагоджено програму, що повністю виконую поставлене індивідуальне завдання та відповідає вимогам. Було отримано і вдосконалено навички у використанні паралельного виконання декількох частин програми.