ЛАБОРАТОРНА РОБОТА № 7

Тема: Багатопоточне програмування в Java

Мета роботи: практика роботи з потоками в Java

Завдання на лабораторну роботу

Завдання 1. Створити консольний Java проект java_lab_7 з пакетом com.education.ztu.

Завдання 2. Створити клас, що розширює Thread:

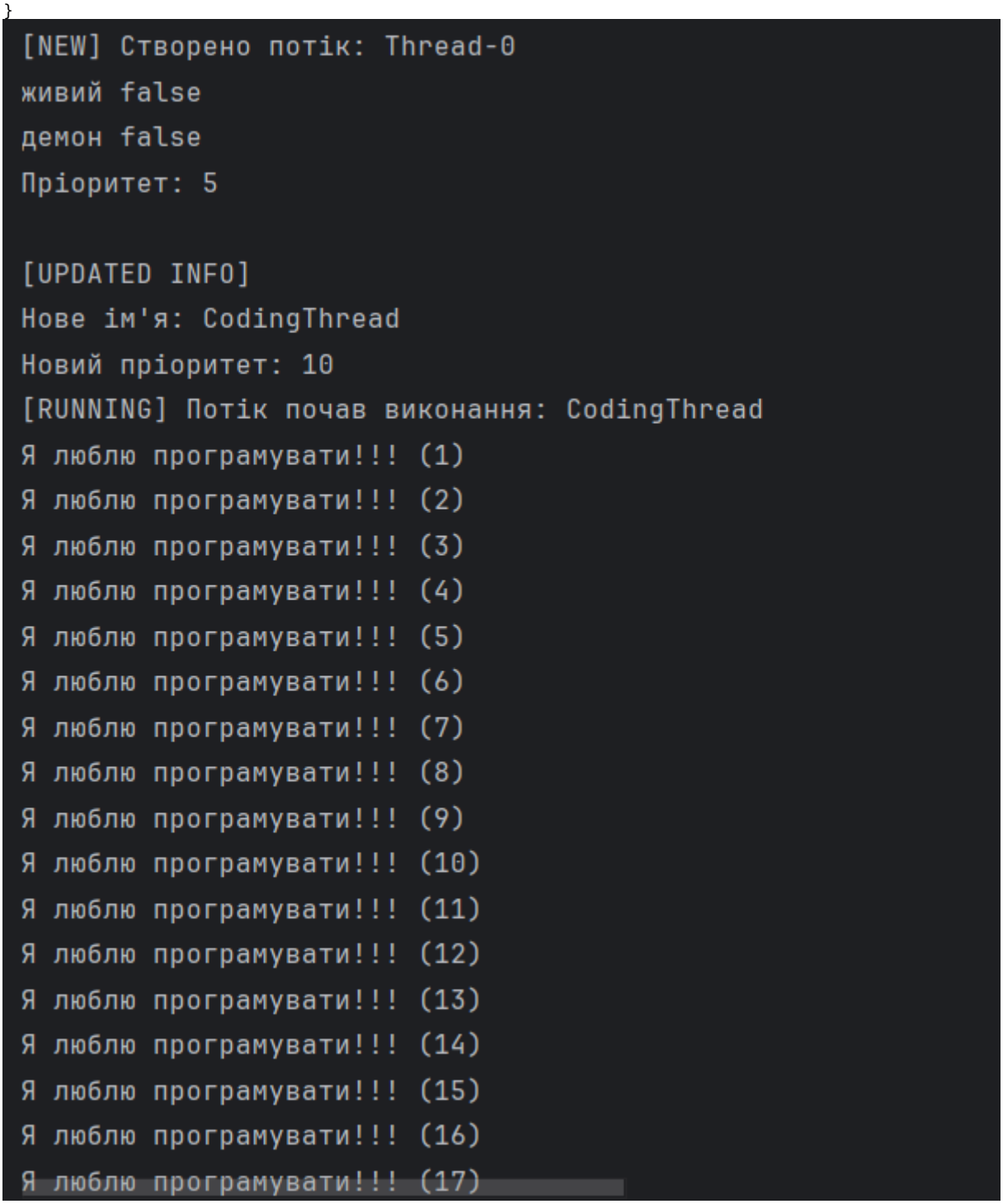
- Створити клас MyThread, що розширює Thread.
- Перевизначити метод run(). У циклі for вивести на консоль повідомлення «Я люблю програмувати!!!» 100 разів.
- Створити екземпляр класу та запустити новий потік.
- Вивести ім'я створеного потоку, його пріорітет, превірити чи він живий, чи є потоком демоном.
- Змінити ім'я, пріорітет створеного потоку та вивести в консоль оновлені значення.
- Після завершення роботи створеного потоку (використати метод join()) вивести ім'я головного потоку, та його пріорітет.
- Відобразити в консолі, коли ваш потік буде в стані NEW, RUNNUNG, TERMINATED.

```
package com.education.ztu;

public class Main {
    public static void main(String[] args) {
        MyThread myThread = new MyThread();
        System.out.println("[NEW] Створено потік: " + myThread.getName());
        System.out.println("живий " + myThread.isAlive());
        System.out.println("демон " + myThread.isDaemon());
        System.out.println("Пріоритет: " + myThread.getPriority());
        myThread.setName("CodingThread");
        myThread.setPriority(Thread.MAX_PRIORITY);
        System.out.println("\n[UPDATED INFO]");
        System.out.println("Нове ім'я: " + myThread.getName());
        System.out.println("Новий пріоритет: " + myThread.getPriority());
        myThread.start();
        try {
            myThread.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        Thread mainThread = Thread.currentThread();
        System.out.println("\n[MAIN THREAD INFO]");
        System.out.println("Ім'я головного потоку: " + mainThread.getName());
        System.out.println("Пріоритет головного потоку: " + mainThread.getPriority());
    }
}
```

```java
package com.education.ztu;

public class MyThread extends Thread {
    @Override
    public void run() {
        System.out.println("[RUNNING] Потік почав виконання: " + getName());
        for (int i = 1; i <= 100; i++) {
            System.out.println("Я люблю програмувати!!! (" + i + ")");
        }
        System.out.println("[TERMINATED] Потік завершився: " + getName());
    }
}
```



```
Я люблю програмувати!!! (83)
Я люблю програмувати!!! (84)
Я люблю програмувати!!! (85)
Я люблю програмувати!!! (86)
Я люблю програмувати!!! (87)
Я люблю програмувати!!! (88)
Я люблю програмувати!!! (89)
Я люблю програмувати!!! (90)
Я люблю програмувати!!! (91)
Я люблю програмувати!!! (92)
Я люблю програмувати!!! (93)
Я люблю програмувати!!! (94)
Я люблю програмувати!!! (95)
Я люблю програмувати!!! (96)
Я люблю програмувати!!! (97)
Я люблю програмувати!!! (98)
Я люблю програмувати!!! (99)
Я люблю програмувати!!! (100)
[TERMINATED] Потік завершився: CodingThread

[MAIN THREAD INFO]
Ім'я головного потоку: main
Пріоритет головного потоку: 5
```

Завдання 3. Створити клас, що реалізує інтерфейс Runnable для виводу в консоль чисел від 0 до 10000, що діляться на 10 без залишку:

- Створити клас MyRunnable, який реалізує інтерфейс Runnable.
- Імплементувати метод run().
- Визначити умову, якщо потік хочуть перервати, то завершити роботу потоку та вивести повідомлення «Розрахунок завершено!!!»
- Створити три потоки, які виконують завдання друку значень.
- Використовуємо статичний метод Thread.sleep(), щоб зробити паузу на 2 секунди для головного потоку, а після цього викликати для створених потоків метод interrupt().

```
package com.education.ztu;

public class Main2 {
    public static void main(String[] args) {
        MyRunnable task = new MyRunnable();
```

```
        Thread t1 = new Thread(task, "Thread-1");
        Thread t2 = new Thread(task, "Thread-2");
        Thread t3 = new Thread(task, "Thread-3");
        t1.start();
        t2.start();
        t3.start();
        try {
            Thread.sleep(2000);
        } catch (InterruptedException e) {
            System.out.println("Головний потік перервано.");
        }
        t1.interrupt();
        t2.interrupt();
        t3.interrupt();
    }
}
```

```
package com.education.ztu;

public class MyRunnable implements Runnable {
    @Override
    public void run() {
        try {
            for (int i = 0; i <= 10000; i++) {
                if (Thread.currentThread().isInterrupted()) {
                    System.out.println("[" + Thread.currentThread().getName() + "] Розрахунок завершено!!!");
                    return;
                }
                if (i % 10 == 0) {
                    System.out.println("[" + Thread.currentThread().getName() + "] " + i);
                }
            }
        } catch (Exception e) {
            System.out.println("[" + Thread.currentThread().getName() + "] Потік зупинено через виключення.");
        }
    }
}
```

```
[Thread-1] 0
[Thread-1] 10
[Thread-1] 20
[Thread-3] 0
[Thread-3] 10
[Thread-3] 20
[Thread-3] 30
[Thread-1] 30
[Thread-1] 40
[Thread-1] 50
[Thread-1] 60
[Thread-2] 0
[Thread-2] 10
[Thread-3] 40
[Thread-3] 50
[Thread-2] 20
[Thread-1] 70
[Thread-3] 60
[Thread-2] 30
[Thread-1] 80
[Thread-3] 70
[Thread-3] 80
[Thread-3] 90
[Thread-2] 40
[Thread-2] 50
```

```
[Thread-3] 9780
[Thread-3] 9790
[Thread-3] 9800
[Thread-3] 9810
[Thread-3] 9820
[Thread-3] 9830
[Thread-3] 9840
[Thread-3] 9850
[Thread-3] 9860
[Thread-3] 9870
[Thread-3] 9880
[Thread-3] 9890
[Thread-3] 9900
[Thread-3] 9910
[Thread-3] 9920
[Thread-3] 9930
[Thread-3] 9940
[Thread-3] 9950
[Thread-3] 9960
[Thread-3] 9970
[Thread-3] 9980
[Thread-3] 9990
[Thread-3] 10000

Process finished with exit code 0
```

Завдання 4. Створити клас, що реалізує інтерфейс Runnable для вививедення арифметичної прогресії від 1 до 100 з кроком 1:

- Створити клас, який реалізує інтерфейс Runnable.
- Створити об'єкт зі статичною змінною result для збереження значення арифметичної прогресії.
- Перевизначити метод run(). Створити цикл for. У циклі виводимо через пробіл значення змінної result. Та додаємо наступне значення до змінної result та чекаємо 0,2 секунду.
- Забезпечити корректну роботу використовуючи синхронізований метод.

- Створити три потоки, які виконують завдання друку значень.

```
package com.education.ztu;

public class Main3 {
    public static void main(String[] args) {
        Runnable task = new Task4();
        Thread t1 = new Thread(task, "Thread-A");
        Thread t2 = new Thread(task, "Thread-B");
        Thread t3 = new Thread(task, "Thread-C");
        t1.start();
        t2.start();
        t3.start();
    }
}
```

```
package com.education.ztu;

public class Task4 implements Runnable {
    private static int result = 1;
    private static final int MAX = 100;
    private static synchronized int getNextValue() {
        if (result <= MAX) {
```

```
            return result++;
        } else {
            return -1;
        }
    }
    @Override
    public void run() {
        while (true) {
            int value = getNextValue();
            if (value == -1) break;
            System.out.println(Thread.currentThread().getName() + ": " + value);
            try {
                Thread.sleep(200);
            } catch (InterruptedException e) {
                System.out.println(Thread.currentThread().getName() + " перервано.");
                break;
            }
        }
    }
}
```

```
Thread-A: 1
Thread-B: 2
Thread-C: 3
Thread-C: 4
Thread-A: 5
Thread-B: 6
Thread-C: 7
Thread-B: 8
Thread-A: 9
Thread-C: 10
Thread-A: 11
Thread-B: 12
Thread-C: 15
Thread-A: 13
Thread-B: 14
Thread-C: 16
Thread-A: 18
Thread-B: 17
```

```
Thread-A: 85
Thread-C: 87
Thread-B: 86
Thread-B: 88
Thread-C: 90
Thread-A: 89
Thread-B: 91
Thread-A: 92
Thread-C: 93
Thread-C: 96
Thread-A: 95
Thread-B: 94
Thread-A: 97
Thread-C: 98
Thread-B: 99
Thread-C: 100

Process finished with exit code 0
```

Завдання 5. Переробити 4 завдання використовуючи блок синхронізації.

```
package com.education.ztu;

public class Main4 {
    public static void main(String[] args) {
        Runnable task = new Task5();

        Thread t1 = new Thread(task, "Thread-1");
        Thread t2 = new Thread(task, "Thread-2");
        Thread t3 = new Thread(task, "Thread-3");

        t1.start();
        t2.start();
        t3.start();
    }
}
```

```
package com.education.ztu;

public class Task5 implements Runnable {
    private static int result = 1;
    private static final int MAX = 100;
    private static final Object lock = new Object();
    @Override
    public void run() {
        while (true) {
            int value;
            synchronized (lock) {
                if (result > MAX) break;
                value = result++;
                System.out.println(Thread.currentThread().getName() + ": " + value);
            }
            try {
                Thread.sleep(200);
            } catch (InterruptedException e) {
                System.out.println(Thread.currentThread().getName() + " перервано.");
                break;
            }
        }
    }
}
```

```
Thread-1: 1
Thread-2: 2
Thread-3: 3
Thread-3: 4
Thread-1: 5
Thread-2: 6
Thread-2: 7
Thread-3: 8
Thread-1: 9
Thread-2: 10
Thread-1: 11
Thread-3: 12
Thread-2: 13
Thread-1: 14
Thread-3: 15
Thread-2: 16
Thread-3: 17
Thread-1: 18
Thread-2: 19
Thread-3: 20
Thread-1: 21
Thread-2: 22
```

```
Thread-1: 81
Thread-2: 82
Thread-3: 83
Thread-1: 84
Thread-2: 85
Thread-1: 86
Thread-3: 87
Thread-2: 88
Thread-1: 89
Thread-3: 90
Thread-1: 91
Thread-3: 92
Thread-2: 93
Thread-3: 94
Thread-2: 95
Thread-1: 96
Thread-3: 97
Thread-2: 98
Thread-1: 99
Thread-3: 100
```

Завдання 6. Створити два потоки Reader та Printer. Reader зчитує введені дані з консолі та записує в змінну. Після цього інформує потік Printer та засипає на 1 секунду, а потік Reader виводить дотриманий рядок. І так повторюється знову, поки користувач не завершить роботу програми.

- Змінну треба використати як об'єкт для синхронізації.
- Тут необхідно використати wait() і notify().

```
package com.education.ztu;

public class SharedData {
    public String message = "";
    public boolean hasNewMessage = false;
}
package com.education.ztu;

import java.util.Scanner;

public class ReaderThread extends Thread {
    private final SharedData data;
    private final Scanner scanner = new Scanner(System.in);
    public ReaderThread(SharedData data) {
        this.data = data;
    }
    @Override
    public void run() {
        while (true) {
            System.out.print("Введіть рядок (або 'exit' для завершення): ");
            String input = scanner.nextLine();

            synchronized (data) {
                data.message = input;
                data.hasNewMessage = true;
                data.notify();
            }

            if ("exit".equalsIgnoreCase(input)) {
                break;
            }

            try {
                Thread.sleep(1000);
```

```
            } catch (InterruptedException e) {
                System.out.println("Reader перервано.");
                break;
            }
        }
    }
}
```

```
package com.education.ztu;

public class PrinterThread extends Thread {
    private final SharedData data;
    public PrinterThread(SharedData data) {
        this.data = data;
    }
    @Override
    public void run() {
        while (true) {
            synchronized (data) {
                while (!data.hasNewMessage) {
                    try {
                        data.wait();
                    } catch (InterruptedException e) {
                        System.out.println("Printer перервано.");
                        return;
                    }
                }
                String msg = data.message;
                data.hasNewMessage = false;
                if ("exit".equalsIgnoreCase(msg)) {
                    System.out.println("Printer завершує роботу.");
                    break;
                }
                System.out.println("Printer: " + msg);
            }
        }
    }
}
```

```
package com.education.ztu;

public class Main5 {
    public static void main(String[] args) {
        SharedData sharedData = new SharedData();
        Thread reader = new ReaderThread(sharedData);
        Thread printer = new PrinterThread(sharedData);
        printer.start();
        reader.start();
    }
}
```

```
Введіть рядок (або 'exit' для завершення): 4
Printer: 4
Введіть рядок (або 'exit' для завершення): 5
Printer: 5
Введіть рядок (або 'exit' для завершення): exit
Printer завершує роботу.
```

Завдання 7. Створити програму для знаходження суми цифр в масиві на 1 000 000 елементів:

- Заповнити масив числами використовуючи клас Random.
- Реалізувати задачу в однопоточному та багатопоточному середовищі.
- Для багатопоточного середовища використати ExecutorService на 5 потоків та об'єкти потоків, що імплементують інтерфейси Runnable або Callable.
- Заміряти час виконання обох варіантів завдання використовуючи System.currentTimeMillis() та вивести результати в консоль.

Завдання 8. *В GitLab проекті Java_labs_ztu, створити директорію Lab_7 та запушити в Lab_7 виконану лабораторну роботу. Надати доступ для перевірки викладачу.*

```
package com.education.ztu;

import java.util.*;
import java.util.concurrent.*;

public class Task7 {
    private static final int ARRAY_SIZE = 1_000_000;
    private static final int THREAD_COUNT = 5;
    public static void main(String[] args) throws InterruptedException, ExecutionException {
        int[] numbers = new int[ARRAY_SIZE];
        Random random = new Random();
        for (int i = 0; i < ARRAY_SIZE; i++) {
            numbers[i] = random.nextInt(10000);
        }
        long startSingle = System.currentTimeMillis();
        long singleThreadSum = sumDigitsSingleThread(numbers);
        long endSingle = System.currentTimeMillis();
        System.out.println("Однопоточна сума цифр: " + singleThreadSum);
        System.out.println("Час (мс): " + (endSingle - startSingle));
        long startMulti = System.currentTimeMillis();
        long multiThreadSum = sumDigitsMultiThread(numbers);
        long endMulti = System.currentTimeMillis();

        System.out.println("Багатопоточна сума цифр: " + multiThreadSum);
        System.out.println("Час (мс): " + (endMulti - startMulti));
    }
    public static long sumDigitsSingleThread(int[] array) {
        long sum = 0;
        for (int num : array) {
            sum += sumDigits(num);
        }
        return sum;
    }
    public static long sumDigitsMultiThread(int[] array) throws InterruptedException, ExecutionException {
        ExecutorService executor = Executors.newFixedThreadPool(THREAD_COUNT);
        List<Future<Long>> futures = new ArrayList<>();
        int chunkSize = array.length / THREAD_COUNT;
        for (int i = 0; i < THREAD_COUNT; i++) {
            int start = i * chunkSize;
            int end = (i == THREAD_COUNT - 1) ? array.length : start + chunkSize;
            futures.add(executor.submit(new DigitSumTask(Arrays.copyOfRange(array, start, end))));
        }
        long totalSum = 0;
        for (Future<Long> future : futures) {
            totalSum += future.get();
        }

        executor.shutdown();
        return totalSum;
    }
    public static int sumDigits(int number) {
        int sum = 0;
```

```
        while (number != 0) {
            sum += number % 10;
            number /= 10;
        }
        return sum;
    }
    static class DigitSumTask implements Callable<Long> {
        private final int[] data;
        public DigitSumTask(int[] data) {
            this.data = data;
        }
        @Override
        public Long call() {
            long sum = 0;
            for (int num : data) {
                sum += Task7.sumDigits(num);
            }
            return sum;
        }
    }
}
```

```
Однопоточна сума цифр: 18005216
Час (мс): 10
Багатопоточна сума цифр: 18005216
Час (мс): 20
```