



Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформаційні систем та технологій

Лабораторна робота №7
Java
Варіант 3 (103)

Виконав:
Студент групи ІА-33
Ничик О. О.

Перевірив:
Лесик В. О.

Мета роботи: набуття практичних навичок у використанні високорівневих методів паралельного виконання та взаємодії потоків, взаємодія потоків на основі умовних змінних, використання пулу потоків, створення додатків з використанням фреймворку Fork/Join.

Хід роботи

Task1

Account.java

```
public class Account {
    private final int id;
    private int balance;

    public Account(int id, int balance) {
        this.id = id;
        this.balance = balance;
    }

    public int getId() { return id; }
    public int getBalance() { return balance; }

    public void withdraw(int amount) { balance -= amount; }
    public void deposit(int amount) { balance += amount; }
}
```

Bank.java

```
public class Bank {

    public void transfer(Account from, Account to, int amount) {

        Account first = from.getId() < to.getId() ? from : to;
        Account second = from.getId() < to.getId() ? to : from;

        synchronized (first) {
            synchronized (second) {

                if (from.getBalance() < amount) return;

                from.withdraw(amount);
                to.deposit(amount);
            }
        }
    }
}
```


Task1Test.java

```
public class Task1Test {

    Run | Debug
    public static void main(String[] args) throws InterruptedException {

        Bank bank = new Bank();
        Random rnd = new Random();

        List<Account> accounts = new ArrayList<>();
        for (int i = 0; i < 200; i++) {
            accounts.add(new Account(i, rnd.nextInt(bound: 10_000)));
        }

        int before = accounts.stream().mapToInt(Account::getBalance).sum();
        System.out.println("Before: " + before);

        int threads = 3000;
        Thread[] workers = new Thread[threads];

        for (int i = 0; i < threads; i++) {
            workers[i] = new Thread(() -> {
                Account a = accounts.get(rnd.nextInt(accounts.size()));
                Account b = accounts.get(rnd.nextInt(accounts.size()));
                if (a == b) return;

                int amount = rnd.nextInt(bound: 100);
                bank.transfer(a, b, amount);
            });
            workers[i].start();
        }

        for (Thread t : workers) t.join();

        int after = accounts.stream().mapToInt(Account::getBalance).sum();
        System.out.println("After: " + after);

        System.out.println(before == after
            ? "OK – гроші збереглись"
            : "ERROR – десь пропали гроші!");
    }
}
```

Task2

RingBuffer.java

```
public class RingBuffer<T> {

    private final Object[] data;
    private int head = 0;
    private int tail = 0;
    private int size = 0;

    public RingBuffer(int capacity) {
        this.data = new Object[capacity];
    }

    public synchronized void put(T value) {
        while (size == data.length) {
            try { wait(); } catch (InterruptedException ignored) {}
        }
        data[tail] = value;
        tail = (tail + 1) % data.length;
        size++;
        notifyAll();
    }

    @SuppressWarnings("unchecked")
    public synchronized T get() {
        while (size == 0) {
            try { wait(); } catch (InterruptedException ignored) {}
        }
        T val = (T) data[head];
        head = (head + 1) % data.length;
        size--;
        notifyAll();
        return val;
    }
}
```

Task2.java

```

public class Task2 {

    Run | Debug
    public static void main(String[] args) throws InterruptedException {

        RingBuffer<String> buffer1 = new RingBuffer<>(capacity: 20);
        RingBuffer<String> buffer2 = new RingBuffer<>(capacity: 20);

        for (int i = 0; i < 5; i++) {
            int id = i;
            Thread producer = new Thread(() -> {
                int msg = 1;
                while (true) {
                    buffer1.put("Потік " + id + " згенерував повідомлення " + msg++);
                }
            });
            producer.setDaemon(on: true);
            producer.start();
        }

        for (int i = 0; i < 2; i++) {
            int id = i;
            Thread worker = new Thread(() -> {
                while (true) {
                    String msg = buffer1.get();
                    buffer2.put("Потік " + id + " переклав: " + msg);
                }
            });
            worker.setDaemon(on: true);
            worker.start();
        }

        for (int i = 0; i < 100; i++) {
            System.out.println(buffer2.get());
        }

        System.out.println(x: "Готово.");
    }
}

```

<https://github.com/iamthegoose/java-fifth-semester/tree/lab9>

Висновок: В ході виконання даної лабораторної роботи я набув практичних навичок у використанні високорівневих методів паралельного виконання та взаємодії потоків, дізнався про взаємодію потоків на основі умовних змінних, використовував пулу потоків, створював додатків з використанням фреймворку Fork/Join..