Звіт
До лабораторної роботи №4
студента гр. КС-21 Кабаченка Артема

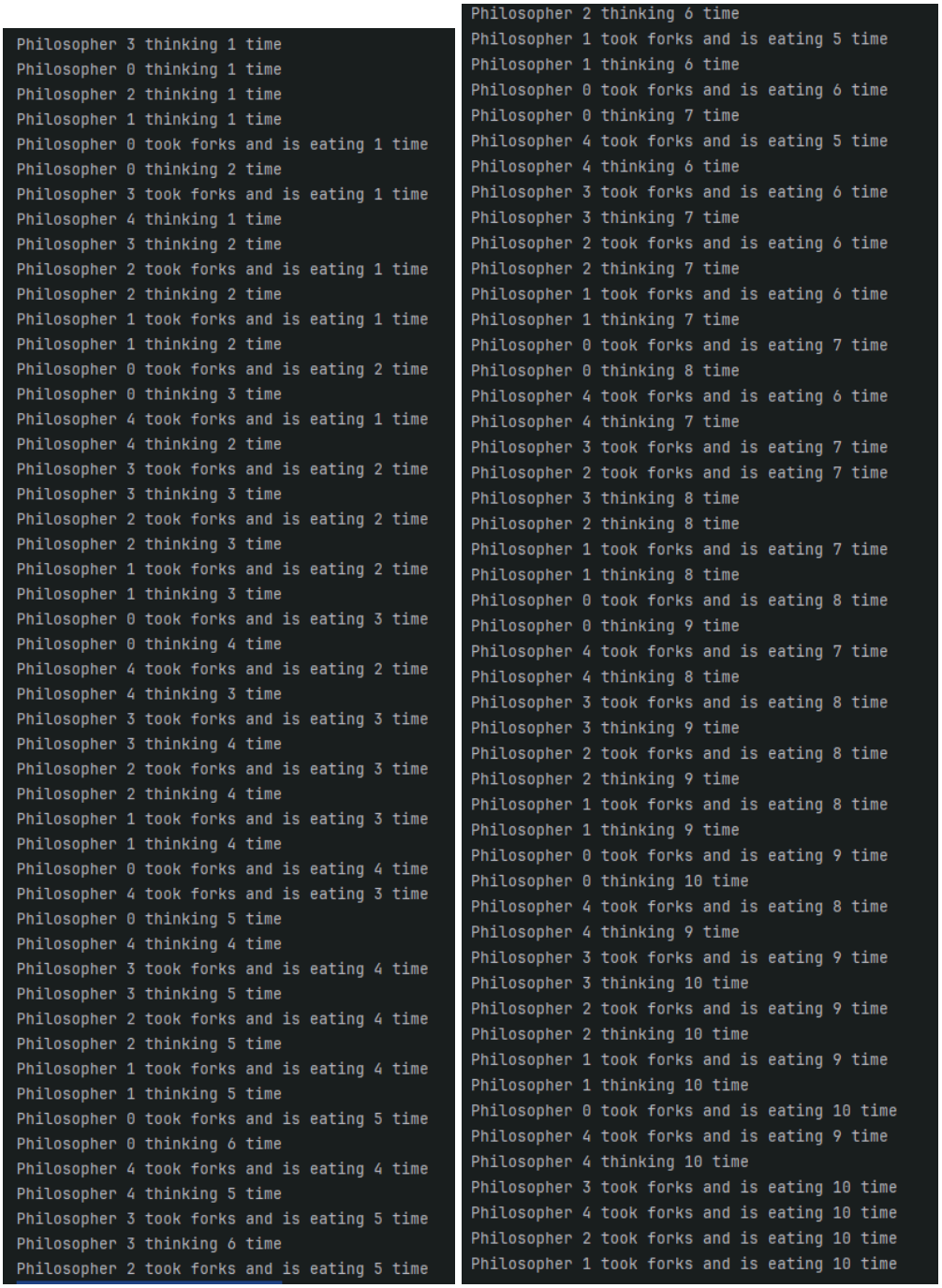
Тема: Взаємне блокування потоків (задача про філософів, що обідають).

Java

```java
class Philosopher extends Thread {  4 usages
    private int id;  3 usages
    private Object leftFork;   // Ліва виделка  2 usages
    private Object rightFork;  // Права виделка  2 usages

    public Philosopher(int id, Object leftFork, Object rightFork) {  1 usage
        this.id = id;
        this.leftFork = leftFork;
        this.rightFork = rightFork;
    }

    // Метод, який описує процес їжі філософа
    private void eat(int times) {  1 usage
        for (int i = 0; i < times; i++) {
            System.out.println("Philosopher " + id + " thinking " + (i + 1) + " time"); // Філософ міркує
            synchronized (leftFork) {  // Забирає ліву виделку
                synchronized (rightFork) { // Забирає праву виделку
                    System.out.println("Philosopher " + id + " took forks and is eating " + (i + 1) + " time"); // Філософ починає їсти
                }
            }
        }
    }

    // Потік філософа
    public void run() {
        eat( times: 10); // Кожен філософ їсть 10 разів
    }
}

public class Main {
    public static void main(String[] args) {
        Object[] forks = new Object[5];  // Масив, що містить виделки

        // Ініціалізація виделок
        for (int i = 0; i < forks.length; i++) {
            forks[i] = new Object();
        }

        Philosopher[] philosophers = new Philosopher[5]; // Масив філософів

        // Ініціалізація філософів
        for (int i = 0; i < philosophers.length; i++) {
            Object leftFork = forks[i]; // Ліва виделка для філософа
            Object rightFork = forks[(i + 1) % forks.length]; // Права виделка для філософа
            philosophers[i] = new Philosopher(i, leftFork, rightFork); // Створення філософа
        }

        // Запуск потоків філософів
        for (Philosopher philosopher : philosophers) {
            philosopher.start();
        }
    }
}

```

1. **Philosophe**: Це клас, який представляє філософа. Кожен філософ володіє своїм ідентифікатором, а також лівою та правою виделкою.
2. **eat**: Це приватний метод, який відображає процес їжі філософа. Філософ спочатку міркує, а потім намагається взяти ліву та праву виделку, щоб почати їсти.

3. **run**: Це метод потоку, який викликається, коли починається виконання потоку. В цьому випадку філософи починають їсти.
4. **Main**: Основний клас програми, де створюються і запускаються філософи.
5. **Масив виделок**: Створюється масив виделок, який використовується для моделювання виделок, які використовують філософи.
6. **Ініціалізація філософів**: Філософи створюються з використанням виделок з масиву. Кожен філософ отримує ліву та праву виделку.
7. **Запуск потоків філософів**: Потоки філософів запускаються, і кожен починає їсти.



Ada

```ada
-- Include necessary Ada packages
with Ada.Text_IO; use Ada.Text_IO;
with GNAT.Semaphores; use GNAT.Semaphores;

-- Define a task type Philosopher
procedure Dinner_Philosophers is
   task type Philosopher is
      entry Start(Id : Integer); -- Entry to start a philosopher
   end Philosopher;

   -- Declare an array of counting semaphores for forks
   Forks : array (1..5) of Counting_Semaphore(1, Default_Ceiling);

   -- Task body for the philosopher
   task body Philosopher is
      Id : Integer;
      Left_Fork, Right_Fork : Integer;
   begin
      -- Accept the starting entry to initialize the philosopher's ID
      accept Start (Id : in Integer) do
         Philosopher.Id := Id;
      end Start;

      -- Assign left and right forks based on philosopher's ID
      Left_Fork := Id;
      Right_Fork := Id rem 5 + 1;

      -- Loop representing the philosopher's activities
      for I in 1..10 loop
         -- Print the philosopher's thinking activity
         Put_Line("Philosopher " & Id'Img & " thinking " & I'Img & " time");

         -- Seize the left and right forks
         Forks(Left_Fork).Seize;
         Forks(Right_Fork).Seize;

         -- Print the philosopher's eating activity
         Put_Line("Philosopher " & Id'Img & " took forks and is eating " & I'Img & " time");

         -- Release the forks after eating
         Forks(Right_Fork).Release;
         Forks(Left_Fork).Release;
      end loop;
   end Philosopher;

   -- Declare an array of philosophers
   Philosophers : array (1..5) of Philosopher;
begin
   -- Start each philosopher
   for I in Philosophers'Range loop
      Philosophers(I).Start(I);
   end loop;

end Dinner_Philosophers;
```

- **Включення необхідних пакетів**:
  - Ada.Text_IO для виводу на консоль.
  - GNAT.Semaphores для роботи з семафорами.
- **Визначення типу задачі "Філософ"**:
  - Визначено тип задачі Philosopher, який має один вхідний вхід Start, що дозволяє почати роботу філософа з вказаним ідентифікатором.
- **Створення масиву семафорів для виділення виделок**:
  - Оголошено масив семафорів Forks для виділення п'яти виделок.

- **Оголошення тіла задачі "Філософ"**:
    - В тілі задачі Philosopher оголошені змінні для ідентифікатора філософа, номерів лівої та правої виделки.
    - Приймається вхід Start, щоб ініціалізувати ідентифікатор філософа.
    - Виконується вибір лівої та правої виделок.
    - Виконується цикл, що відображає діяльність філософа: думання, взяття виделок та їжа.
- **Оголошення масиву філософів та запуск кожного з них**:
    - Оголошено масив Philosophers з п'ятьма філософами.
    - Запускається кожен філософ у циклі, передаючи йому його ідентифікатор.