**Лабораторна робота №4**

Шевченко Н. В.

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

**Мета:** Познайомитись з основними причинами виникнення взаємного блокування паралельних потоків, розібрати способи уникнення блокування потоків на прикладі задачі про філософів, що обідають.

**Ada**

Цей код реалізує проблему обідувальників, що їдять в умовах використання семафорів для уникнення взаємного блокування.

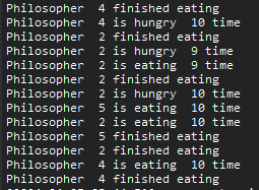
Починається програма з оголошення двох семафорів: My\_Semaphore і Room\_Semaphore. My\_Semaphore є захищеним типом з методами Acquire, Release і Get\_Count, які використовуються для отримання, звільнення та отримання значення лічильника семафора відповідно. Room\_Semaphore є семафором, що рахує доступність місць в кімнаті для одночасної їжі.

Після оголошення семафорів створюється масив Fork\_Semaphores, що представляє семафори для кожної виделки, індексований від 1 до 5. Крім того, оголошується масив Philosophers, який буде містити об'єкти задач Philosopher\_Arbitrator.

Philosopher\_Arbitrator - це задача, яка представляє філософа. У своєму тілі задача отримує свій ідентифікатор, вибирає ліву та праву виделку згідно зі своїм ідентифікатором, і потім зациклюється 10 разів, щоб виконати свою діяльність. Під час кожного циклу філософ намагається поїсти.

У місцевому циклі філософ намагається захопити доступ до кімнати, а потім перевіряє, чи доступні обидві виделки. Якщо обидві доступні, він захоплює їх, інакше він відпускає кімнату і спробує ще раз. Після того, як філософ захопив обидві виделки, він починає обід і виводить повідомлення про це.

Після закінчення обіду філософ виводить повідомлення, вільно відпускає виделки і повертається до початку циклу для наступного обіду.



**Java**

**Room**

Цей код реалізує проблему обідувальників за допомогою семафорів для уникнення голодування та взаємного блокування.

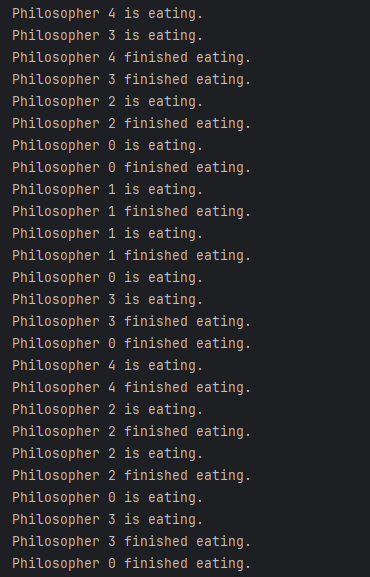
У методі PhilosopherRoomStart створюються філософи та виделки. Для кожного філософа створюється по дві виделки, що є семафорами, та столова за допомогою семафора, щоб обмежити кількість філософів, які можуть одночасно сидіти за столом.

Кожна виделка є об'єктом класу ForkSem, який містить семафор. Філософи представлені об'єктами класу PhilosopherRoom, які реалізують інтерфейс Runnable для виконання в окремих потоках.

У методі run філософи зациклюються, спочатку чекаючи 1 секунду, а потім намагаються зайняти стіл, взяти виделки, поїсти та відпустити виделки. Це повторюється 100 разів для кожного філософа.

Кожен філософ намагається зайняти стіл за допомогою семафора diningRoom. Після цього філософ намагається взяти обидві виделки, якщо вони доступні, і викликає метод pickUpForks, щоб поїсти. Після того, як він закінчить їсти, він відпускає виделки та звільняє стіл.

Клас ForkSem відповідає за кожну виделку. У ньому ініціалізується семафор з одним дозволом. Метод pickUp забирає дозвіл на виделку, а putDown відпускає її.



**Philosopher**

Цей код ілюструє ситуацію з обідом філософів і використовує синхронізацію потоків за допомогою моніторів (синхронізованих методів та блоків) для уникнення взаємоблокування.

У методі createPhilosophers створюються філософи та їх виделки. Кожному філософу присвоюється дві виделки: ліва та права. Філософи представлені об'єктами класу Philosopher, які розширюють клас Thread для виконання в окремих потоках. Метод повертає масив філософів.

Клас Philosopher представляє філософа. Він має унікальний ідентифікатор, ліву та праву виделку. У методі run філософ перебуває у циклі, в якому він думає, вживає їжу та повторює це 100 разів. Кожен цикл філософ здійснює послідовний процес: думає, намагається взяти обидві виделки, їсть та кладе їх на стіл.

Клас Fork представляє виделку. Він має унікальний ідентифікатор та відображає, чи знаходиться виделка на столі. Методи take та put синхронізовані для уникнення взаємоблокування. Філософ, який намагається взяти виделку, чекає, доки вона не буде на столі, а потім бере її. Після того, як філософ закінчив їсти, він кладе виделку на стіл.



**Висновок**

У лабораторній роботі було розглянуто проблему філософів, що обідають, як приклад виникнення взаємного блокування потоків. Представлено реалізації на Ada та Java з використанням семафорів та моніторів для синхронізації доступу до спільних ресурсів (виделок та місць в їдальні) та уникнення взаємоблокування. Продемонстровано різні підходи до вирішення цієї проблеми в паралельному програмуванні.