Laboratorium V – Teoria współbieżności

Weronika Szybińska, 21.11.2022

Treść zadania:

1. Zaimplementować trywialne rozwiązanie z symetrycznymi filozofami. Zaobserwować problem blokady.
2. Zaimplementować rozwiązanie z widelcami podnoszonymi jednocześnie.
3. Zaimplementować rozwiązanie z lokajem.
4. Wykonać pomiary dla każdego rozwiązania i wywnioskować co ma wpływ na wydajność każdego rozwiązania

Opis rozwiązania:

Do rozwiązania zadania 1, wykorzystana została klasa ReentrantLock. Na początku tworzonych jest 5 obiektów klasy Widelec oraz 5 obiektów klasy Filozof. Widelec posiada dwie funkcje podnies() oraz odloz(), które z wykorzystaniem ReentrantLock powodują zablokowanie widelca na czas jedzenia. Filozofowie w przypadku, gdy widelec po ich lewej stronie jest wolny podnoszą go i nie odkładają póki nie dostaną dostępu do prawego widelca. Poniżej przedstawiona jest implementacja danych klas.

```
class Filozof extends Thread {
    private int _licznik = 0;
    private final Widelec leftFork;
    private final Widelec rightFork;
    private boolean leftForkInPossesion = false;
    private final Random rand = new Random();

    public Filozof(Widelec leftFork, Widelec rightFork){
        this.leftFork = leftFork;
        this.rightFork = rightFork;
    }

    public void run() {
        while (true) {
            while(!leftForkInPossesion){
                leftForkInPossesion  = leftFork.podnies();
            }
            while(leftForkInPossesion){
                if(rightFork.podnies()){
                    leftForkInPossesion = false;
                    try{
                        sleep(rand.nextInt( bound: 100));

                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                }
            }
            ++_licznik;
            if (_licznik % 2 == 0) {
                System.out.println("Filozof: " + Thread.currentThread() +
                        "jadlem " + _licznik + " razy");
            }
            leftFork.odloz();
            rightFork.odloz();
        }
    }
}
```

```java
class Widelec {
    private final ReentrantLock forkLock = new ReentrantLock();
    public boolean podnies() {
        return forkLock.tryLock();
    }
    public void odloz() { forkLock.unlock(); }
}
```

2. W przypadku zadnia drugiego zmodyfikowana została klasa Filozof. Obiekty tej klasy zamiast blokować lewy widelec czekając aż prawy się zwolni, próbują podnieść oba widelce naraz. Poniżej przedstawiona zmodyfikowana klasa.

```java
class Filozof extends Thread {
    private int _licznik = 0;
    private final Widelec leftFork;
    private final Widelec rightFork;
    private final Random rand = new Random();

    public Filozof(Widelec leftFork, Widelec rightFork){
        this.leftFork = leftFork;
        this.rightFork = rightFork;
    }
    public void run() {
        while (true) {
            while(true){
                boolean leftForksInPossesion = leftFork.podnies();
                boolean rightForksInPossesion = rightFork.podnies();
                if(leftForksInPossesion && rightForksInPossesion){
                    break;
                }
                else if(leftForksInPossesion){
                    leftFork.odloz();
                }
                else if(rightForksInPossesion){
                    rightFork.odloz();
                }
            }
            try{
                sleep(rand.nextInt( bound: 100));
            } catch (InterruptedException e) {
                        e.printStackTrace();
            }
            ++_licznik;
            if (_licznik % 10 == 0) {
                System.out.println("Filozof: " + Thread.currentThread() +
                        "jadlem " + _licznik + " razy");
            }
            leftFork.odloz();
            rightFork.odloz();
        }
    }
```

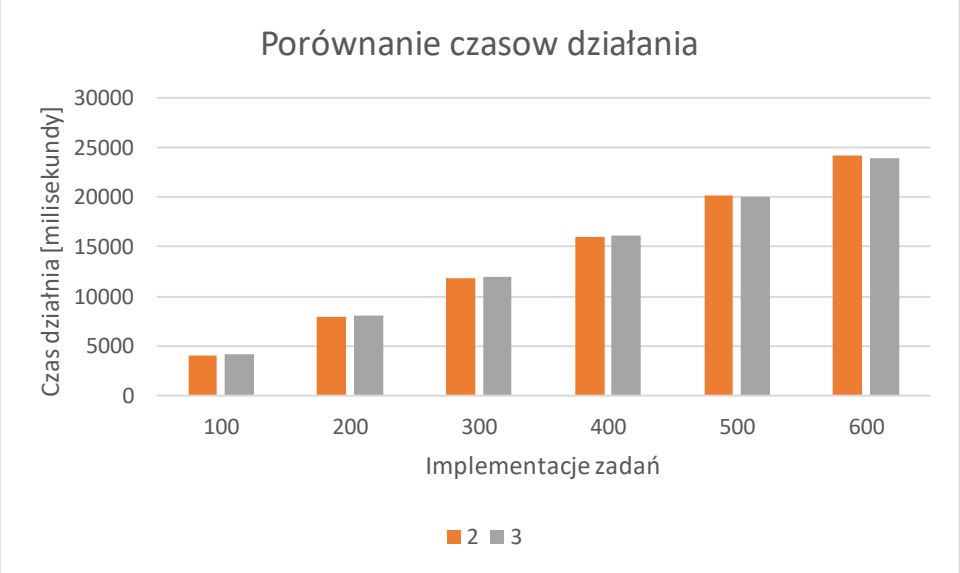
3. W przypadku zadania 3 wykorzystany został dodatkowo Semafor o wartości początkowej 4. Imituje on lokaja, który dopuszcza tylko 4 filozofów do jedzenia. Jest on przekazywany jako jeden a argumentów konstruktora.

```
class Filozof extends Thread {
    private int _licznik = 0;
    private final Widelec leftFork;
    private final Widelec rightFork;
    private Semaphore flunkey;
    private final Random rand = new Random();

    public Filozof(Widelec leftFork, Widelec rightFork, Semaphore flunkey)
        this.leftFork = leftFork;
        this.rightFork = rightFork;
        this.flunkey = flunkey;
    }
```

```
    public void run() {
        while (true) {
            try {
                flunkey.acquire();
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            while(true){
                boolean leftForksInPossesion = leftFork.podnies();
                boolean rightForksInPossesion = rightFork.podnies();
                if(leftForksInPossesion && rightForksInPossesion){
                    break;
                }
                else if(leftForksInPossesion){
                    leftFork.odloz();
                }
                else if(rightForksInPossesion){
                    rightFork.odloz();
                }
            }
            try{
                sleep(rand.nextInt( bound: 100));
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            ++_licznik;
            if (_licznik % 100 == 0) {
                System.out.println("Filozof: " + Thread.currentThread() +
                        "jadlem " + _licznik + " razy");
            }
            leftFork.odloz();
            rightFork.odloz();
            flunkey.release();
        }
    }
}
```

4. Na koniec zostały wykonane pomiary. W przypadku programu z zadania 1, pomiary nie zostały wykonane, gdyż po około 20 obejściach, program się blokował. W przypadku implementacji zadani 2 i 3, zostały zmierzone czasy działania programu w zależności od wartości licznika jaką miały osiągnąć wszystkie obiekty klasy Filozof.



WNIOSKI:

W przypadku implementacji zadania 1, wystąpiła blokada. Dzieję się tak w przypadku, gdy wszyscy filozofie trzymają swój lewy widelec i czekają na zwolnienie się prawego. Z podpunktu 4 można wywnioskować, że implementacje 2 i 3 nie różnią się zbytnio czasem działania. Wnioski te jednak są błędne, gdyż program z zadania 3 powinien być szybszy, gdyż w zadaniu 2 występować może zjawisko zagłodzenia. Możliwe, że różnice czasowe są bardziej widoczne w przypadku ustawienia bardzo dużej wartości zmiennej licznik.

BIBLIOGRAFIA:

http://galaxy.agh.edu.pl/~balis/dydakt/tw/lab8/tw-5fil.pdf

http://aragorn.pb.bialystok.pl/~wkwedlo/OS1-4.pdf

https://pl.wikipedia.org/wiki/Problem_ucztuj%C4%85cych_filozof%C3%B3w