Sprawozdanie nr 1 - Tomasz Szewczyk

Wyścig

Problem

Głównym tematem zajęć było wprowadzenie w temat współbieżności oraz opis implementacji problemu w maszynie wirtualnej języka Java. Współbierznośc może być realizowana przez procesy lub wątki. Wątki różnią się od procesów między innymi współdzieleniem pamięci: wątki uruchamiają się w obrębie procesów i współdzielą stertę, natomiast każdy proces na swój stos oraz stertę. Java implementuje współbierzność za pomocą wątków. Umożliwia to wykonywanie wielu operacji równocześnie. Na przykład kiedy jeden wątek oczekuje na wykonanie operacji wejścia/wyjścia, inny wciąż może wykonywać obliczenia. Dzięki wykorzystaniu wątków możemy lepiej wykorzystać dostępne zasoby komputera.

Istnieje wiele sytuacji w których należy skorzystać ze współbierzności. Należą do nich: złożone obliczenia, które można rozdystrybuować na wiele fizycznych procesorów, obsługa interfejsów użytkownika, wykonywanie blokujących operacji.

Wątki w języku Java tworzy się na dwa sposoby. Poprzez kasę Thread lub implementację interfejsu Runnable.

Zadanie

Na zajęciach należało zaprezentować problem wyścigu, to jest sytuacji w której więcej niż jeden wątek korzysta jednocześnie z jednego zasobu i próbuje zmienić jego stan. Zajście takiej sytuacji powoduje, że program zachowuje się w sposób nieprzewidywalny.

Implementacja

```
package tomaszszewczyk.lab1;

public class lab1 {
    public static void main(String[] args) throws InterruptedException {
        final long startTime = System.currentTimeMillis();
        final Counter myCounter = new Counter();

        Thread firstThread = new Thread() {
            public void run() {
                for (int i = 0; i < 1000000; i++) {
                    myCounter.inc();
                }
            }
```

```
        };

        Thread secondThread = new Thread() {
            public void run() {
                for (int i = 0; i < 1000000; i++) {
                    myCounter.dec();
                }
            }
        };

        firstThread.start();
        secondThread.start();
        secondThread.join();
        firstThread.join();
        final long endTime = System.currentTimeMillis();
        System.out.println(myCounter.getVal());
        System.out.println(endTime - startTime);
    }
}

class Counter extends Thread {
    private int val;

    Counter() {
        this.val = 0;
    }

    void inc() {
        this.val++;
    }

    void dec() {
        this.val--;
    }

    int getVal() {
        return val;
    }
}
```

Wynik

Zgodnie z przewidywaniami uruchomienie kodu daje inny wynik za każdym razem:

```
$ ./lab1
-14326
 18

$ ./lab1
-12559
 17

$ ./lab1
-197496
 16
```