

Programowanie obiektowe II

Laboratorium nr 13

Stefan Sokołowski

Zadanie 1:

W programie

iis.ans-elblag.pl/~stefan/Dydaktyka/ProgObiII/Prog13/Dom/Z1_Nawiasy.java

klasa `Nawias` opisuje wątek, którego działanie polega na

- wypisaniu otwarcia `otw`; następnie
- wykonaniu działania wątku `wt`; w końcu
- wypisaniu zamknięcia `zam`.

Program główny (w klasie `Z1_Nawiasy`) uruchomi więc wątek `wt2`, który w trakcie swojego działania uruchomi wątek `wt1`, który uruchomi wątek `wt0`. Dlatego mogłoby się wydawać, że program wypisze

```
begin {[]}' end
      |
      v
    wt0
      |
      v
    wt1
      |
      v
    wt2
```

Jednak on tego nie robi (proszę sprawdzić). Wyjaśnić, dlaczego tak się dzieje. Poprawić klasę `Nawias` (tylko ją!) tak, żeby program zachowywał się zgodnie z oczekiwaniem.

Wskazówka: komenda `t.join()`; powoduje zawieszenie działania wątku, w którym została umieszczona, do czasu, gdy zakończy działanie wątek `t`.

Zadanie 2:

W programie

iis.ans-elblag.pl/~stefan/Dydaktyka/ProgObiII/Prog13/Dom/Z2_Licznik.java

liczniki (obiekty klasy `Licznik`) można odczytywać metodą `get()` albo zwiększać metodą `up()`. Wątek z klasy `WGore` zwiększa używany przez siebie licznik 100 tys. razy — więc program główny powinien wydrukować liczbę 100000. Jednak nie robi tego: drukuje 0.

Proszę wyjaśnić, dlaczego tak się dzieje.

Poprawić program tak, żeby istotnie drukował 100000, modyfikując **wyłącznie** komendy synchronizujące wątki.

Zadanie 3:

Program główny w zad. 2, poprawiony zgodnie z wymaganiami tego zadania, zmienić w taki sposób, żeby zamiast jednego wątku `WGore` stworzyć trzy takie wątki związane z tym samym licznikiem 11:

```
Thread t1 = new Thread(new WGore(11));
Thread t2 = new Thread(new WGore(11));
Thread t3 = new Thread(new WGore(11));
```

i wystartować je wszystkie (jednak nie likwidując współbieżności ich działania):

```
t1.start(); t2.start(); t3.start();
```

Ponieważ działają one na tym samym liczniku, powinny wspólnie zwiększyć jego wartość do 300 tys.

W moich testach program drukował różne liczby za każdym wywołaniem, ale do 300 tys. zawsze dużo brakowało. Wyjaśnić dlaczego. Sprawdzić, co się stanie, jeśli metodę `up()` zwiększającą licznik zadeklarujemy jako „synchronizowaną” czyli niepodzielną:

```
public synchronized void up() { wartosc = wartosc+1; }
```

Zadanie 4:

W programie

iis.ans-elblog.pl/~stefan/Dydaktyka/ProgObiII/Prog13/Dom/Z3_Slowa.java

rozgałęziające się drzewo wątków współpracuje w celu wypisania wszystkich słów możliwych do utworzenia z podanej liczby liter i mających podaną długość. Np. dla wartości

```
public static int wlkAlfabetu=2, dlugSlowa=3;
```

są to wszystkie słowa długości 3 zbudowane z 2 pierwszych liter alfabetu:

```
AAA ABA BAA BBA AAB ABB BAB BBB
```

W głównej klasie `Z3_Slowa` dokonuje się pomiar czasu działania programu. Jednak wynik tego pomiaru zostaje wydrukowany zanim jeszcze zostaną wygenerowane słowa i oczywiście nie odnosi się do czasu działania, a tylko do czasu startowania całej maszynerii.

Proszę umieścić w tym programie komendy synchronizujące wątki w taki sposób,

- żeby pomiar czasu dotyczył całego działania programu (i drukował się po jego zakończeniu);
- żeby wątki nadal działały maksymalnie współbieżnie.

Nie trzeba dbać o to, żeby słowa wypisały się w kolejności alfabetycznej, ale wszystkie muszą zostać wypisane.

Oczywiście program ma działać poprawnie dla dowolnych wartości stałych `wlkAlfabetu` i `dlugSlowa`.

Zadanie 5:

Program

iis.ans-elblog.pl/~stefan/Dydaktyka/ProgObiII/Prog13/Dom/Z4_ProdKons.java

powinien działać następująco:

- Producent generuje losowe liczby rzeczywiste (w zmiennej `Stale.ile` określona jest ich ilość) i wkłada kolejno do magazynu (metoda `Magazyn.wloz()`). Następnie informuje konsumenta o nowej liczbie i „odpoczywa” (liczbę milisekund określa liczba `Stale.czasProd`).
- Konsument sprawdza w pętli, czy magazyn jest pusty. Jeśli tak, to zasypia; a jeśli nie, to wyjmuje liczbę z magazynu (metoda `Magazyn.wyjmij()`) i wkłada ją do swojej własnej listy od razu na takie miejsce, żeby ta lista pozostawała uporządkowana. Następnie „odpoczywa” (liczbę milisekund określa liczba `Stale.czasKons`).

- Za każdym obrotem pętli Producenta drukowana jest literka P; więc po zakończeniu działania programu tych literek powinno być `Stale.ile`. Za każdym obrotem pętli Konsumenta drukowana jest literka K; po zakończeniu działania programu tych literek może być więcej, bo w przypadku natrafienia na pusty magazyn, potrzebny jest następny obrót pętli.

Ale ten program nie działa z powodu braku komend synchronizujących. Proszę wpisać właściwe komendy we właściwe miejsca, żeby program zadziałał. Proszę przy tym tak to zrobić, żeby nie ograniczyć współbieżności; w pewnym zakresie można tym sterować przez dopasowywanie czasów zwłoki w klasie `Stale`.

O tym, jak bardzo wątki działają współbieżnie, świadczy rozkład literek P i K. Jeśli na końcu wydruku znajdzie się dużo literek K, to znaczy, że Producent jest szybszy od Konsumenta; jeśli po każdej P następują dwie K, to Konsument jest szybszy od Producenta. Zależy nam na tym, żeby oba wątki działały mniej więcej tak samo szybko.

Gotowy program proszę przetestować dla różnych wartości zmiennej `Stale.ile`.
