Paradygmaty programowania - ćwiczenia
Lista 11

Na wykładzie 11 była mowa o niskopoziomowych mechanizmach JVM. Poprawne działanie tych mechanizmów wynika z warunków, sformułowanych w modelu pamięci Javy (JMM = Java Memory Model). https://docs.oracle.com/javase/specs/jls/se8/html/jls-17.html#jls-17.4
Poniżej przytoczone są dosłowne cytaty z książki:
B.Goetz, T.Peierls, J.Bloch, J.Bowbeer, D.Holmes, D.Lee,
Java. Współbieżność dla praktyków, Helion, Gliwice 2007, rozdział 16
Mowa jest w nich o Javie, ale te uwagi odnoszą się również do języka Scala, ponieważ wykorzystywany jest ten sam model pamięci.

Przypuśćmy, że jeden z wątków przypisuje wartość zmiennej aVariable:

```
aVariable = 3;
```

Model pamięci zajmuje się odpowiedzią na pytanie: „w jakich okolicznościach wątek, który odczytuje aVariable, zobaczy wartość 3?”. Pytanie wydaje się głupie, ale przy braku synchronizacji jest wiele powodów, dla których może z opóźnieniem (jeśli w ogóle) zobaczyć wynik działania innego wątku. Kompilatory mogą generować instrukcje w innej kolejności, niż wynikałoby to z kodu źródłowego, lub przechowywać dane w rejestrach procesora zamiast w pamięci; procesory mogą wykonywać instrukcje równolegle lub poza kolejnością; pamięci podręczne modyfikują kolejność, w której zapisy do pamięci trafiają do głównej pamięci komputera; wartości zawarte w lokalnej pamięci jednego procesora mogą nie być widoczne dla innych procesorów. […]
Specyfikacja języka Java wymaga, by maszyna wirtualna **zachowywała w wątku semantykę warunków**. […]
JMM określa minimalne gwarancje, jakie maszyna wirtualna musi zapewnić, by zapis zmiennej w jednym wątku był widoczny przez inne wątki. […]

Wygodnym (do użytku we własnej głowie) modelem wykonywania programów jest wyobrażenie sobie, iż istnieje jedna kolejność, w której operacje wykonują się w programie, niezależnie od tego, które procesory go wykonują, i że każdy odczyt zmiennej widzi ostatni zapis w kolejności wykonywania przeprowadzony przez dowolny z procesorów. Ten przyjemny, choć mało realistyczny, model nosi nazwę **spójności sekwencyjnej** (ang. sequential consistency). Twórcy oprogramowania często błędnie zakładają spójność sekwencyjną, ale żaden nowoczesny system wieloprocesorowy jej nie zapewnia. Nie zapewnia jej również JMM. Klasyczny model obliczeń sekwencyjnych, model von Neumanna, jest tylko ogólnym przybliżeniem sposobu działania nowoczesnych systemów wieloprocesorowych. […]

JMM definiuje porządek częściowy nazywany **zdarzyło się wcześniej** (ang. happened-before) dla wszystkich akcji w programie. Jeśli akcja A zdarza się wcześniej niż akcja B, to wątek, wykonujący akcję B widzi zmiany pamięci, wykonane przez akcję A (nawet jeśli A jest wykonywana w innym wątku). Jeśli akcje A i B nie są w porządku **zdarzyło się wcześniej**, to maszyna wirtualna może dowolnie zmieniać kolejność ich wykonania (patrz wykład 11, str. 9-11).

Paradygmaty programowania - ćwiczenia
Lista 11

Wszystkie programy mają być napisane w języku Java.

1. Przeanalizuj poniższy program (z folderu zad1). Dwa wątki zwiększają 200 000 razy wspólny licznik (egzemplarz klasy IntCell) o 1. Po ich zakończeniu wartość licznika powinna oczywiście wynosić 400 000. Uruchom ten program kilka razy. Jak wyjaśnisz otrzymane wyniki?

```
class IntCell {
  private int n = 0;
   public int getN() {return n;}
   public void setN(int n) {this.n = n;}
}

class Count extends Thread {
  private  static IntCell n = new IntCell();

   @Override   public void run() {
    int temp;
    for (int i = 0; i < 200000; i++) {
      temp = n.getN();
      n.setN(temp + 1);
    }
  }

   public static void main(String[] args) {
    Count p = new Count();
    Count q = new Count();
    p.start();
    q.start();
    try { p.join(); q.join(); }
    catch (InterruptedException e) { }
    System.out.println("The value of n is " + n.getN());
  }
}
```

2. a) Popraw powyższy program, wykorzystując mechanizm monitorów.
   b) Popraw powyższy program, wykorzystując mechanizm semaforów.

3. W folderze zad3 znajduje się aplikacja, rozwiązująca problem producent/konsument dla trzech producentów i trzech konsumentów.
a) Przeanalizuj ten program. Należy umieć wyjaśnić jego działanie! (wystarczy ustnie).
b) Czy użycie metody notify() zamiast notifyAll() byłoby bezpieczne w tym programie? Odpowiedź uzasadnij.

4. Problem ucztujących filozofów
Problem polega na zsynchronizowaniu działań pięciu chińskich filozofów, którzy mieszkają w położonej na odludziu chacie (według innych przekazów była to wieża z kości słoniowej), mieszczącej jadalnię i salę do medytacji. Żyją oni, starając się zachować równowagę między yang i yin, Niebem i Ziemią, duchem i ciałem. Filozofowie są na tyle zaawansowani w ascezie, że nie muszą spać ani wykonywać innych przyziemnych czynności (rzecz dzieje się u kresu dziesięciu mitycznych epok, prawdopodobnie w czasie panowania Żółtego Cesarza Huang–ti, kiedy to wiele rzeczy było możliwych, jak o tym świadczą zachowane legendy). Czas upływa im na myśleniu i medytacji w przeznaczonej do tego sali, muszą jednak od czasu do czasu jeść (zgodnie z klasyczną zasadą zachowania równowagi yin–yang). W jadalni stoi okrągły stół z pięcioma miseczkami, między którymi leży pięć pałeczek do ryżu, a na środku stołu znajduje się duża misa z niewyczerpalnym zapasem gotowanego ryżu - swego rodzaju róg (chińskiej) Amaltei . Etykieta mówi, że każdy filozof siada zawsze na tej samej poduszce, je zawsze ze swojej miseczki i może korzystać tylko z pałeczek znajdujących się po obu jej stronach. Filozofowie przestrzegają ustalonej etykiety jak prawdziwi konfucjaniści, na przykład nigdy nie sięgnęliby po ryż dłonią. Nie urządzają jednak żadnych zebrań, nie uznają hierarchii, nie posiadają żadnej informacji, dotyczącej spraw przyziemnych. Kiedy poczują głód to spontanicznie ruszają do jadalni.

Jak powinni oni odprawiać swój rytuał, aby nie umrzeć z głodu? Rozwiązanie powinno spełniać następujące warunki:

1. Filozof je tylko wtedy, gdy ma dwie pałeczki.
2. Dwóch filozofów nie może jednocześnie trzymać tej samej pałeczki.
3. Nie występuje blokada (sytuacja patowa). Może ona wystąpić np. wtedy, gdy wszyscy filozofowie podniosą lewe pałeczki i będą czekać na zwolnienie prawych.
4. Nikt nie może być zagłodzony. Oczywista z pozoru strategia, polegająca na poczekaniu, aż obie pałeczki będą wolne, może spowodować zagłodzenie dwóch filozofów (dlaczego?).
5. Żaden z filozofów nie zajmuje się tylko jedzeniem. Po zakończeniu posiłku każdy odkłada pałeczki i wraca do sali medytacji.
6. Filozofowie podnoszą i odkładają pałeczki po jednej naraz.
7. Nie można wyróżniać żadnego z filozofów (algorytmy ich działania powinny być takie same).

Jedno z rozwiązań wymaga zaangażowanie odźwiernego, pilnującego drzwi jadalni i pozwalającego przebywać w niej jednocześnie co najwyżej czterem filozofom. Dzięki temu co najmniej dwom filozofom, siedzącym przy stole, brakuje co najmniej jednego sąsiada, a zatem co najmniej jeden filozof może jeść (dlaczego?). Rozwiązanie problemu pięciu filozofów wygląda więc następująco. Na początku wszyscy filozofowie medytują w przeznaczonej do tego sali. Kiedy któryś z nich poczuje głód, wstaje i kieruje się do jadalni. Jeśli w jadalni jest mniej niż czterech filozofów, odźwierny przepuszcza go. W przeciwnym razie filozof siada u drzwi jadalni, czekając na znak odźwiernego. W jadalni filozof zajmuje swoje stałe miejsce i próbuje podnieść najpierw lewą pałeczkę. Jeśli jest ona używana przez lewego sąsiada, czeka na jej zwolnienie, po czym analogicznie stara się podnieść prawą. Mając obie pałeczki filozof zaczyna się posilać. Po nasyceniu głodu odkłada najpierw lewą, a następnie prawą pałeczkę, wstaje i bezszelestnie wraca na swoje miejsce w sali medytacji. Jeśli u progu jadalni są oczekujący filozofowie, odźwierny wskazuje na jednego z nich, pozwalając mu wejść do jadalni. Wybory odźwiernego są uczciwe, co znaczy, że żaden z oczekujących nie będzie pomijany w nieskończoność.

Edsger W. Dijkstra, który w 1971 roku opublikował tę starożytną historię, dostosował ją do realiów Zachodu, m.in. każąc filozofom jeść spaghetti dwoma widelcami. Próbując rozwiązać problem pięciu filozofów, wymyślił on mechanizm *semaforów*. Jak widać, znajomość mitologii (i to nie tylko grecko–rzymskiej) może być bardzo inspirująca.

Napisz program, rozwiązujący problem ucztujących filozofów za pomocą semaforów.
Przedstaw filozofów jako procesy, sekcją krytyczną jest jedzenie, a zasobami dzielonymi są pałeczki do ryżu. Procesy są ponumerowane od 0 do 4, co odpowiada stałym miejscom filozofów przy stole i wykonują się współbieżnie. Użycie każdej pałeczki jest kontrolowane przez semafor binarny, a odźwierny jest reprezentowany przez semafor ogólny z wartością początkową 4.