Teoria Współbieżności
Ćwiczenie 1

1 Cel ćwiczenia

Celem ćwiczenia jest zapoznanie studentów z wątkami oraz synchronizacją w Java.

2 Wątki a procesy

1. Rodzaje wątków
   (a) Wątki użytkownika (user thread)
   (b) Wątki systemowe (kernel thread)
   (c) LWP - Lightweight Process (np. w Solaris)
   (d) Procesory fizyczne
2. Przydział wątków do procesorów
   (a) Przetwarzanie równoległe
   (b) Przetwarzanie współbieżne
   (c) Wielozadaniowość kooperacyjna
   (d) Wielozadanoiwość z wywłaszczaniem (preemptive)

3 Wątki w Javie

1. Tworzenie wątków
2. Klasa Thread
3. Interfejs *Runnable*

4. API JavaDoc

5. Anonimowa klasa wewnętrzna:

```
new Thread() {
  public void run() {
    for(;;) System.out.println("Thread 1");
  }
}.start();
new Thread() {
  public void run() {
    for(;;) System.out.println("Thread 2");
  }
}.start();
```

4 Wątki w JVM

1. Architektura JVM: Chapter 5 of Inside the Java Virtual Machine by Bill Venners.

   - Obszary danych:
     - sterta
     - stos
     - przestrzeń metod
     - rejestr licznika programu
   - Obszary współdzielone między wątkami oraz prywatne

2. Cykl życia wątku, stany wątku

   - New
   - Runnable
   - Running
   - Not Running
   - Dead

3. Szeregowanie wątkow w Javie

   (a) Ogólne pojęcia szeregowania: wywłaszczania (*preemption*), podziału czasu (*time-slicing*, *time-sharing*), priorytety.

(b) Dokładne zachowanie się wątków jest zależne od platformy.

(c) Wątki mogą być zaimplementowane całkowicie w przestrzeni użytkownika lub korzystać z natywnych interfejsów platformy.

(d) Wątkom można przypisać priorytety (1-10). Jedyną gwarancją jest to, że wątkom o najwyższym priorytecie zostanie przydzielony CPU. Wątki o niższym priorytecie mają gwarancję przydziału CPU tylko wtedy, gdy wątki z wyższym priorytetem są zablokowane, w przeciwnym wypadku nie ma tej gwarancji.

(e) Specyfikacja nie zakłada podziału czasu.

(f) Operacje odczytu i zapisu danych typów prostych (primitives) pomiędzy pamięcią główną a roboczą pamięcią wątku są atomowe. Jedynym wyjątkiem mogą być operacje na 64-bitowych typach long i double, które mogą być zrealizowane jako dwie operacje 32-bitowe.

(g) Reguły szeregowania

- W każdym momencie, spośród kilku wątkow w stanie RUNNABLE wybierany jest ten o najwyższym priorytecie
- Wykonywany wątek może być wywłaszczony, jeśli pojawi się wątek o wyższym priorytecie, gotowy do wykonania
- Jeśli wiele wątków ma ten sam priorytet, wybierany jest jeden z nich, wg kolejności (round-robin)
- Na niektórych systemach może być zaimplementowany podział czasu (wątki sa wywłaszczane po upływie kwantu czasu)

5 Wyścig

1. Więcej niż jeden wątek korzysta jednocześnie z zasobu dzielonego, przy czym co najmniej jeden próbuje go zmienić.
2. Przyczyna niedeterministycznego zachowania się programu.
3. Może prowadzić do trudnych do wykrycia błędów.
4. Pojęcie *thread-safety* (bezpieczeństwo dla wątków, wielobieżność).

6 Ćwiczenie

1. Proszę zaimplementować klasę *Counter* oraz 2 wątki: inkrementujący i dekrementujący współdzielony licznik.
2. Proszę przetestować działanie programu (czas, wynik) na różnych systemach operacyjnych oraz wersjach JVM, z brakiem oraz z użyciem synchronizacji (słowo kluczowe *synchronized*).
3. Proszę sprawdzić ile wątków da się utworzyć w systemie.

Literatura

[1] Bill Venners, Inside the Java Virtual Machine (rozdział 5, rozdział 20), McGraw-Hill Companies; 2nd Bk&Cdr edition, 2000.

[2] Bruce Eckel, "Thinking in Java" - rozdział o wątkach

[3] Bartosz Baliś, Teoria Współbieżności, materiały własne