Kurs programowania

Wykład 8

Wojciech Macyna

Wątki

Definicja

Wątek (ang. thread) – część programu wykonywana współbieżnie w obrębie jednego procesu; w jednym procesie może istnieć wiele wątków.

Problem: Wspólne dane

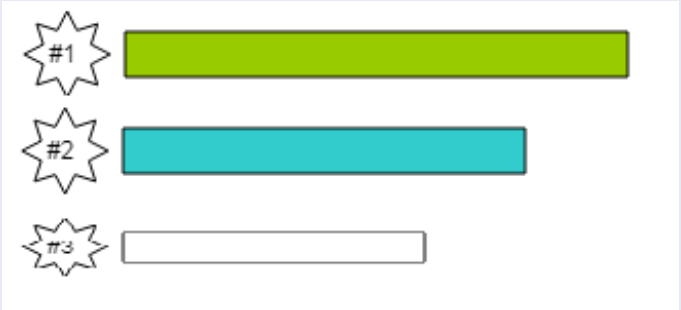
Wątki korzystają z tych samych danych. Mówi się, że wątki współdzielą przestrzeń adresową. Oznacza to tyle, że obiekty dostępne dla jednego wątku są widoczne także w innych wątkach

Klasa `Thread` jest klasą bazową wątków. Jej konstruktor jako argument przyjmuje klasę z zaimplementowanym interfejsem `Runnable` a uruchamia wątek metodą `start()` która wywołuje metodę `run()`. Wątki mogą też być definiowane jako pochodne klasy `Thread`.

Program bez wątków, jeden procesor, trzy zadania

Program z wątkami, jeden procesor, trzy zadania

Procesory wielordzeniowe, każde zadanie na osobnym procesorze



Trzy zadania uruchomione w wątkach na dwóch procesorach

Przykład

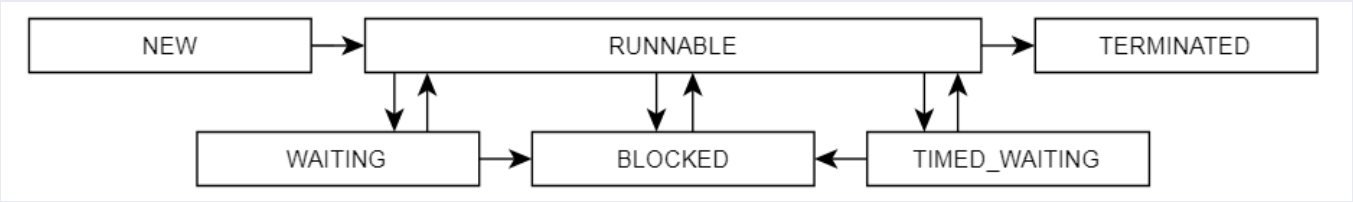
Przykłady działania dwóch niezsynchronizowanych wątków

`RunnableDemo.java` – z pętlą nieskończoną

`RunnableDemo1.java` – z pętlą skończoną

`RunnableDemo2.java` – zatrzymanie wątku

Diagram stanów wątków



Stany wątku

- NEW – nowy wątek, który nie został jeszcze uruchomiony,
- RUNNABLE – wątek, który może wykonywać swój kod,
- TERMINATED – wątek, który zakończył swoje działanie,
- BLOCKED – wątek zablokowany, oczekujący na zwolnienie współdzielonego zasobu,
- WAITING – wątek uśpiony,
- TIMEDWAITING – wątek uśpiony na określony czas.

Klasa `Thread`

Usypianie wątku – `Thread.sleep(int ms)`

Zawieszenie wykonywania wątku. Metoda wywołuje wyjątek `InterruptedException` i musi być umieszczona w środowisku `try`.

Próba przełączenia wątku – `Thread.yield()`

Sugerowanie zarządcy wątków, że w tym momencie można przełączyć sterowanie na inny wątek (o tym samym priorytecie).

Przerwanie wątku – `Thread.interrupt()`

Jeśli wątek jest w stanie BLOCKED lub WAITING (wywołane są metody sleep() oraz wait(), wówczas wywołanie interrupt() powoduje przerwanie tego stanu i wyrzucenie wyjątku `InterruptedException`. Przykład: `Interrupt.java`.
Jeśli wątek nie jest w stanie BLOCKED lub WAITING, wówczas wywołanie interrupt() nie powoduje przerwania wątku, ale flaga przerwania wątku jest ustawiona na `true`. Przykład: `Interrupt1.java`.

Klasa `Thread`

Priorytet wątku – `getPriority()` i `setPriority(int)`

Wątki mogą mieć ustawione priorytety pomiędzy `Thread.MIN_PRIORITY` a `Thread.MAX_PRIORITY`. Wątki o wyższym priorytecie są wykonywane w pierwszej kolejności co może spowodować, ze wątki o niższym priorytecie nie będą wykonywane. Wątki są startowane z domyślnym priorytetem: `Thread.NORM_PRIORITY`.

Wątki-demony – `setDaemon()`

Demon to wątek który działa w tle programu. Program nie czeka np. z zakończeniem działania na wątki które są demonami. Ustawienie wątku jako demona musi się odbyć przed jego uruchomieniem.

Przykład

`Priority1.java`

Klasa `Thread`

Łączenie wątków – `join()`

Czeka aż zakończy się wykonywanie wątku. Dopiero potem rozpoczynają swoje działanie następne wątki.

Nadawanie nazwy – `setName(string name)`

Nadanie nazwy która jest między innymi wyświetlana przez metodę `toString()` obiektu.

Przykład

`JoinTest.java`

Współdzielenie zasobów

Wątki z punktu widzenia programu są wykonywane niezależnie i nie wiadomo w jakiej kolejności zostaną wywołane. Stąd przy korzystaniu ze wspólnych zasobów musimy zapewnić im pewną synchronizację i zapobiec kolizjom. JAVA ma wbudowany mechanizm zapobiegający takim sytuacjom.

Słowo kluczowe `synchronized`

Za pomocą tego słowa oznaczamy metodę która nie powinna być przerwana ze względu na możliwość kolizji. Metoda ta zawsze będzie wykonywana w całości bez względu na możliwość przejścia na inny wątek.

Środowisko `synchronized(object){...}`

Powiązanie nieprzerywalnego ciągu komend z obiektem który jest traktowany jako semafor.

Współdziałanie wątków

Metody `wait()` i `notifyAll()`

Metoda `sleep()` nie zwalnia blokad. Metoda `wait(int ms)` zawiesza wykonywanie wątku na `ms` milisekund i zwalnia blokady. Jej działanie kończy się w wyniku upłynięcia czasu, albo wywołania metod `notify()` lub `notifyAll()`. `notify()` budzi pierwszy pierwszy wątek który wywołał `wait()`.`notifyAll()` budzi wszystkie wątki, które wywołały `wait()`. Będzie działał ten o najwyższym priorytecie. Dla `wait()` bez argumentów nie ma warunku czasowego.

Zakleszczenie

Niewłaściwe stosowanie metody `wait()` przez różne wątki może doprowadzić do zakleszczenia, czyli blokady.

Przykłady

`Synch1.java` – brak synchronizacji

`Synch2.java` – zastosowanie metody `synchronized`

`Synch3.java` – zastosowanie metody `synchronized` oraz `wait()` i `notifyAll()`

`Synch4.java` – zastosowanie środowiska `synchronized` oraz `sleep()`

`Synch5.java` – zastosowanie środowiska `synchronized` oraz `wait()` i `notify()`

`Synch6.java` – przykład zakleszczenia