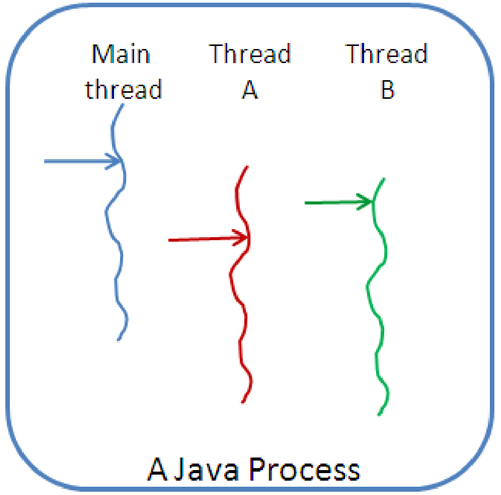
Programowanie wielowątkowe

Michał Korwel

Wątki - wprowadzenie i przykład

Tworząc swoje aplikacje, szczególnie te wykorzystujące interfejs użytkownika z pewnością spotkałeś się z sytuacją, w której pewna czynność, jak na przykład obliczenie wyniku skomplikowanej funkcji, czy pobranie pewnych danych z bazy danych, zabierało dużo czasu, a przez to aplikacja sprawiała wrażenie jakby się zawiesiła. Nie jest to pożądana funkcjonalność i każdy programista chciałby jej uniknąć.
Istnieje jednak rozwiązanie, które w stosunkowo prosty sposób pozwala poradzić sobie z tym problemem – są to **wątki**.
Wątki pozwalają na symultaniczne wykonywanie pewnych operacji dzięki czemu czas wykonania pewnych operacji można znacząco skrócić. W przypadku przykładu z „zawieszeniem” się interfejsu użytkownika można pewne skomplikowane obliczenia wykonać **asynchronicznie** w tle, dzięki czemu użytkownik aplikacji będzie miał lepsze odczucia w związku z jej użytkowaniem. Na obrazku wygląda to tak:



Wątki - wprowadzenie i przykład

To co istotne to fakt, że **zastosowanie wątków sprawdza się nawet na procesorze, który posiada tylko jeden rdzeń**. Jest to spowodowane tym, że każdy z wątków może otrzymywać swój czas procesora na wykonanie pewnych operacji. W przypadku jednego wątku nie ma wyjścia – jeśli wchodzimy do funkcji obliczającej skomplikowane równanie, cały interfejs użytkownika jest zamrażany aż do momentu skończenia obliczeń, natomiast jeśli obliczenia uruchomimy w wątku niezależnym od interfejsu będą one naprzemiennie otrzymywały krótki czas procesora i będą sprawiały wrażenie wykonywania się równoległego (a w przypadku procesora wielordzeniowego faktycznie tak będą działały) – oczywiście pomijamy tu fakt tego, że obok nich wykonuje się wiele innych procesów, które również walczą o uzyskanie czasu procesora. Bardzo ważne jest również to, że jeśli uruchomimy jeden po drugim np. 10 wątków, to wcale nie będą one wykonywały się sekwencyjnie jeden po drugim, jeśli sami o to nie zadbamy. Jeśli jakieś zadanie nie zdąży się wykonać w czasie, który został dla niego przydzielony, to może ono zostać przerwane na pewien czas.

Wątki - wprowadzenie i przykład

Kiedy wykorzystywać wątki?

W wielu sytuacjach:

- Wszelkie obliczenia, które mogą zablokować interfejs użytkownika powinny być wykonywane asynchronicznie
- Animacje, które powinny być przetwarzane niezależnie od interfejsu użytkownika
- Pobieranie danych z internetu (zamiast przetwarzać strony internetowe jedna po drugiej można połączyć się np. z 10 jednocześnie)
- W ogólności wszystkie operacje wejścia/wyjścia, zapis i odczyt plików, czy baz danych
- Złożone obliczenia, które mogą być podzielone na mniejsze podzadania

Mając za sobą ten krótki wstęp teoretyczny przejdźmy więc do rzeczy bardziej praktycznych.

Wątki - wprowadzenie i przykład

Wątki w Javie można tworzyć na kilka sposobów, poprzez:

- jawne rozszerzenie klasy Thread
- stworzenie klasy implementującej interfejs Runnable, który może być wykonany w osobnym wątku (Thread)
- stworzenie klasy implementującej interfejs Callable, który może być wykonany w osobnym wątku (Thread)

Preferowane jest stosowanie rozszerzeń interfejsów (czyli 2 i 3 punkt), ponieważ dają one dużo lepszą elastyczność, szczególnie jeśli dojdziemy do momentu szeregowania wątków, utrzymywania stałej puli wątków wykonujących się w tle. Interfejsy Runnable i Callable są do siebie bardzo podobne, jednak najważniejszą różnicą jest to, że Callable może zwrócić w wyniku pewną wartość, natomiast w przypadku Runnable nie ma takiej możliwości.

Napiszmy prostą aplikację, która utworzy 10 wątków, z których każdy będzie miał za zadanie jedynie wyświetlenie swojego przypisanego ID, a następnie wstrzymanie swojego działania na krótki okres czasu – i tak w kółko.

Wątki - wprowadzenie i przykład

MyRun.java - klasa pozwalająca obiekt, który będzie wykonywany w osobnym wątku

```
public class MyRun implements Runnable {

    private int id;

    public MyRun(int id) {
        this.id = id;
    }

    @Override
    public void run() {
        while(true) {
            System.out.println("Watek "+id);
            try {
                //usypiamy wątek na 100 milisekund
                Thread.sleep(100);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
```

Jak widać interfejs Runnable posiada jedną metodę, którą musimy zaimplementować- **run()**. Wszystko co się w niej znajduje zostanie wykonane po uruchomieniu wątku, do którego przekażemy obiekt klasy MyRun. My utworzyliśmy wewnątrz niej nieskończoną pętlę, której zadaniem jest wyświetlanie ID wątku, a następnie wstrzymanie działania na 100 milisekund za pomocą statycznej metody **sleep()**.

Wątki - wprowadzenie i przykład

Runner.java - klasa testowa

```
public class Runner {
    public static void main(String[] args) {
        Runnable[] runners = new Runnable[10];
        Thread[] threads = new Thread[10];

        for(int i=0; i<10; i++) {
            runners[i] = new MyRun(i);
        }

        for(int i=0; i<10; i++) {
            threads[i] = new Thread(runners[i]);
        }

        for(int i=0; i<10; i++) {
            threads[i].start();
        }
    }
  }
```

W klasie Runner utworzyliśmy tablicę typu Runnable (do której przypiszemy nasze obiekty MyRun) oraz drugą tablicę obiektów Thread - czyli tabilcę, która będzie przechowywała obiekty wątków, które będziemy mogli uruchomić.

W pierwszej pętli for tworzymy obiektu typu MyRun. Obiekty te to jednak nie są jeszcze obiekty wątków - te tworzymy w drugiej pętli, jak widać w celu utworzenia obiektu Thread przekazujemy w konstruktorze nasze wcześniej utworzone obiekty MyRun (rozszerzające Runnable).

W trzeciej pętli nie pozostaje nam nic innego jak uruchomienie wątki poprzez wywołanie metody **start()** - powoduje ona rozpoczęcie wykonania kodu, który stworzyliśmy w metodzie **run()**.

Wątki - wprowadzenie i przykład

Wynik:

```
Watek 0
  Watek 3
  Watek 1
  Watek 7
  Watek 5
  Watek 2
  ...
```

Co ciekawe, jeśli spojrzymy na wydruk programu, to może on być po chwili nieco zaskakujący, ponieważ numery wątków nie będą wyświetlały się w kolejności tworzenia obiektów, ale w kolejności losowej - jest to związane z tym o czym pisałem we wprowadzeniu - możliwości losowego przypisywania czasu procesora różnym wątkom.

Wątki - wprowadzenie i przykład

Dla porównania zobaczmy co się stanie, jeśli zupełnie pominiemy kwestię wątków w przypadku naszego kodu.

```
public class MyRun {

    private int id;

    public MyRun(int id) {
        this.id = id;
    }

    public void run() {
        while(true) {
            System.out.println("NieWatek "+id);
            try {
                //usypiamy wątek na 100 milisekund
                Thread.sleep(100);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
  }
```

Wątki - wprowadzenie i przykład

Tym razem nie implementujemy interfejsu Runnable - tworzymy jedynie nieskończoną pętlę wewnątrz metody run().

```
public class Runner {
    public static void main(String[] args) {
        MyRun[] notRunners = new MyRun[10];

        for(int i=0; i<10; i++) {
            notRunners[i] = new MyRun(i);
        }

        for(int i=0; i<10; i++) {
            notRunners[i].run();
        }
    }
  }
```

Tworzymy tablicę obiektów MyRun, przypisujemy do niej obiekty, a następnie wywołujemy metodę run(). Jak się jednak okazuje w konsoli eclipse wyświetli nam się jedynie "Watek 0", co jest spowodowane tym, że kod wykonywany jest w tym przypadku sekwencyjnie - wchodząc do pętli pierwszego obiektu nasza aplikacja się w niej zawiesza i wykonuje bez końca, tak więc metoda run() obiektu o id=1 nie wykona się nigdy.

Wątki - wprowadzenie i przykład

Wynik:

```
NieWatek 0
  NieWatek 0
  NieWatek 0
  NieWatek 0
  ...
```

Wątki – dopełnienie wiedzy

Na koniec zachęcam do zapoznania się z dokumentacją klas: Thread, Runnable. W kolejnych lekcjach mam nadzieję poruszyć tematy nieco bardziej zaawansowane takie jak synchronizacja, zarządzanie wątkami, utrzymywanie stałej puli wątków, czy ich wykorzystanie w połączeniu z klasą Robot do pisania aplikacji automatyzujących pewne czynności, czy tworzenia botów.

Materiały i filmy, które polecam w ramach dopełnienia wiedzy na temat wątków w javie:

https://www.youtube.com/watch?v=2wEJLjppwFY

https://docs.oracle.com/javase/7/docs/api/java/util/concurrent/ExecutorService.html

https://docs.oracle.com/javase/tutorial/essential/concurrency/executors.html

Zadania z wątkami dla chętnych :

https://docs.oracle.com/javase/tutorial/essential/concurrency/QandE/questions.html

Tutaj wyślij swoje CV : office@ttpsc.pl