

Java Threads

Tworzenie wątków

Wątek jest obiektem klasy `Thread` (lub klasy pochodnej).

Dziedziczenie z klasy `Thread`

Etapy tworzenia wątku:

zdefiniowanie klasy `OurThread` wywiedzionej z klasy `Thread`

implementacja metody `run()`, która zawiera program wątku

utworzenie obiektu `oth` (lub obiektów) zdefiniowanej klasy `OurThread`

uruchomienie wątku

wywołanie metody `start()` utworzonego obiektu `oth`.

```
public class OurThread extends Thread {  
    :  
    public void run() {  
        :  
    }  
    :  
}
```

```
// w celu uruchomienia wątku  
OurThread oth = new OurThread();  
oth.start();
```

Implementacja interfejsu `Runnable`

Etapy tworzenia wątku

zdefiniowanie klasy `OurClass` implementującej interfejs `Runnable`

implementacja metody `run()`, która zawiera program wątku

utworzenie obiektu `oc` zdefiniowanej klasy `OurClass`

utworzenie obiektu `th` (lub obiektów) klasy `Thread` z przekazaniem obiektu `oc`, jako aktualnego parametru konstruktora

uruchomienie wątku

wywołanie metody `start()` utworzonego obiektu `th` klasy `Thread`

```
public class OurClass implements Runnable {  
    :  
    public void run() {  
        :  
    }  
    :  
}
```

```
// w celu uruchomienia wątku  
OurClass oc = new OurClass();  
Thread th = new Thread(oc);  
th.start();
```

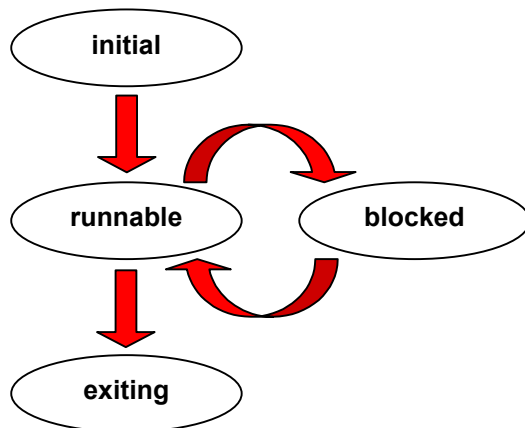
Definicja interfejsu Runnable:

```
public interface Runnable {  
    public abstract void run();  
}
```

Definicja metody run() klasy Thread:

```
public void run() {  
    if (target != null)  
        target.run();  
}
```

Stany wątku



Java Threading API

void start() — uruchamia wątek

void stop() — kończy działanie wątku

void run() — metoda wykonywana przez wątek (główny program wątku)

void join([long milsec [, int nanosec]]) — oczekuje na zakończenia wątku (można podać czas oczekiwania)

boolean isAlive() — sprawdza, czy wątek działa

Metody statyczne

static Thread currentThread() — zwraca obiekt reprezentujący aktualnie wykonywany wątek

static int enumerate(Thread threadArray[]) — zwraca obiekty reprezentujące wszystkie wątki procesu

static int activeCount() — zwraca liczbę aktywnych wątków procesu

static void sleep(long milsec [, int nanosec]) — usypia wątek na podany okres czasu

Nadawanie nazw wątkom

void setName(String name) —

String getName() —

Konstruktory

Thread()

Thread(Runnable target)

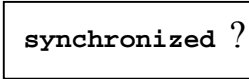
Thread(String name)

Thread(Runnable target, String name)

Synchronizacja wątków — wzajemne wykluczanie

Metoda typu *synchronized* (synchronizacja na obiekcie)

```
public class Account {  
    private float total;  
    public synchronized boolean deduct(float t) {  
        if (t <= total) {  
            total -= t;  
            return true;  
        }  
        return false;  
    }  
    public void add(float t) {  
        total += t;  
    }  
}
```



Metoda typu *synchronized* zajmuje zamek związany z obiektem, będzie zatem wykluczać wykonanie innych metod typu *synchronized* na tym obiekcie.

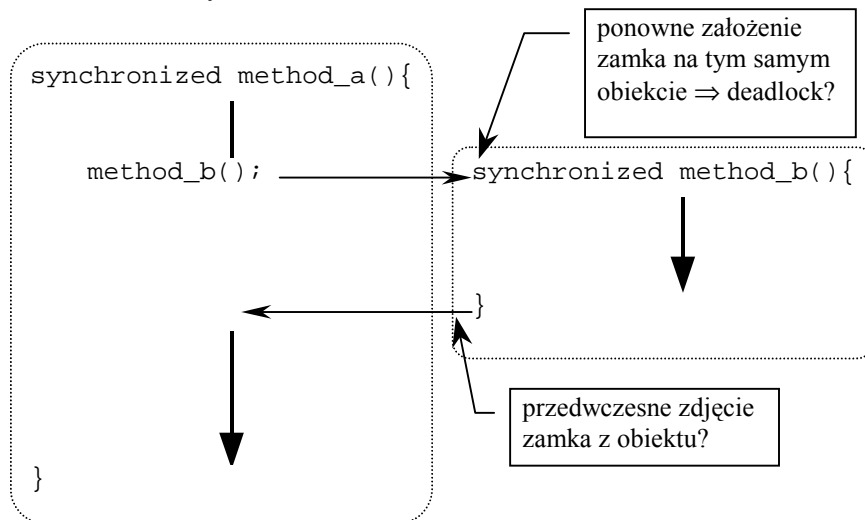
Blok typu *synchronized*

```
synchronized (object) {  
    :  
}
```

Jeżeli tylko fragment kodu metody ma się wykluczać z innymi metodami typu *synchronized*, można to osiągnąć w poniższy sposób:

```
class OurClass {  
    public a_method(...) {  
        :  
        synchronized (this) {  
            :  
        }  
        :  
    }  
}
```

Problem: co się stanie, jeśli metoda typu `synchronized` zostanie wywołana z innej metody typu `synchronized` (czyli przez ten sam wątek). Czy będzie deadlock, a jeśli nie, to czy nie nastąpi przedwczesne zwolnienie blokady obiektu.



Synchronizacja w metodach statycznych

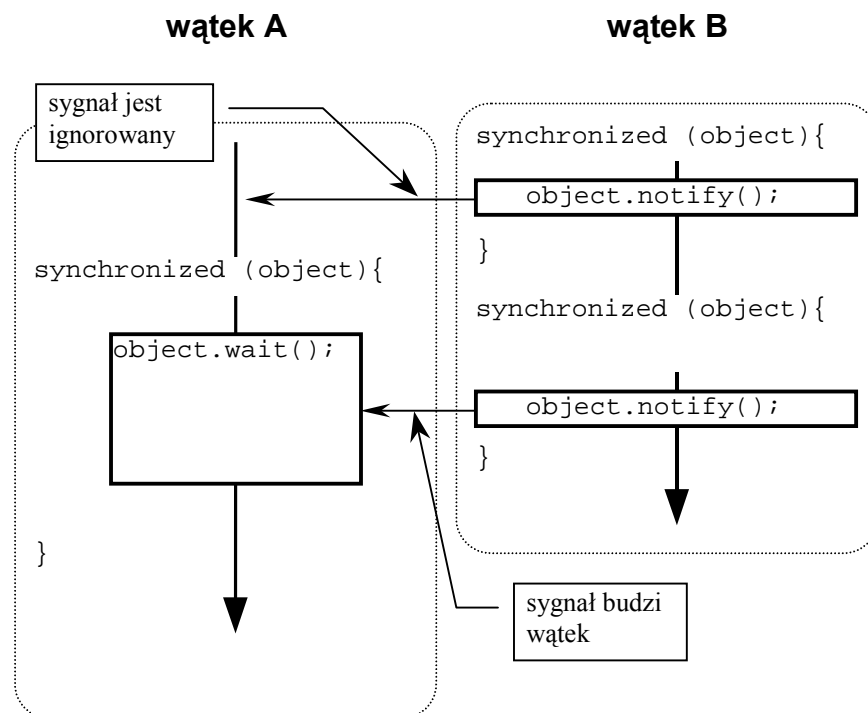
Usypianie i budzenie (`wait()`, `notify()` i `notifyAll()`)

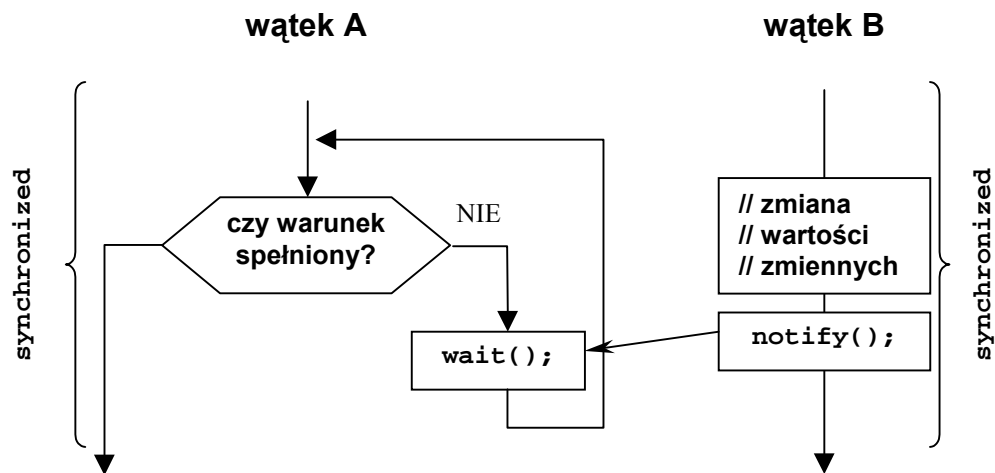
`void wait([long milsec [, int nanosec]])` — czeka na spełnienie warunku (na sygnał wysyłany przez `notify()` lub `notifyAll()`)

`void notify()` — wysyła sygnał do wątku oczekującego po wywołaniu metody `wait()` danego obiektu

`void notifyAll()` — wysyła sygnał do wszystkich wątków oczekujących po wywołaniu metody `wait()`

Metody `wait()` i `notify()` (`notifyAll()`) powinny być wywoływane w metodzie lub bloku `synchronized`.





Szeregowanie wątków

Ustawianie priorytetów wątków

Metody dostępne w obiekcie klasy Thread:

`void setPriority(int priority)` — ustawia priorytet wątku
`int getPriority()` — zwraca priorytet wątku

Zawieszanie i wznowianie wątków

Metody dostępne w obiekcie klasy Thread:

`void suspend()` — zawiesza wątek (wątek nie zwalnia blokad)
`void resume()` — wznowia wykonywanie zawieszonego wątku

Metoda dostępna w klasie Thread:

`static void yield()` — „oddaje procesor” innemu wątkowi o tym samym priorytecie

Demony

Demon jest takim wątkiem, który kończy swoje działanie po zakończeniu ostatniego wątku użytkownika

`void setDaemon(boolean on)` — zmienia wątek użytkownika na wątek-demon
`boolean isDaemon()` — sprawdza, czy wątek jest demonem