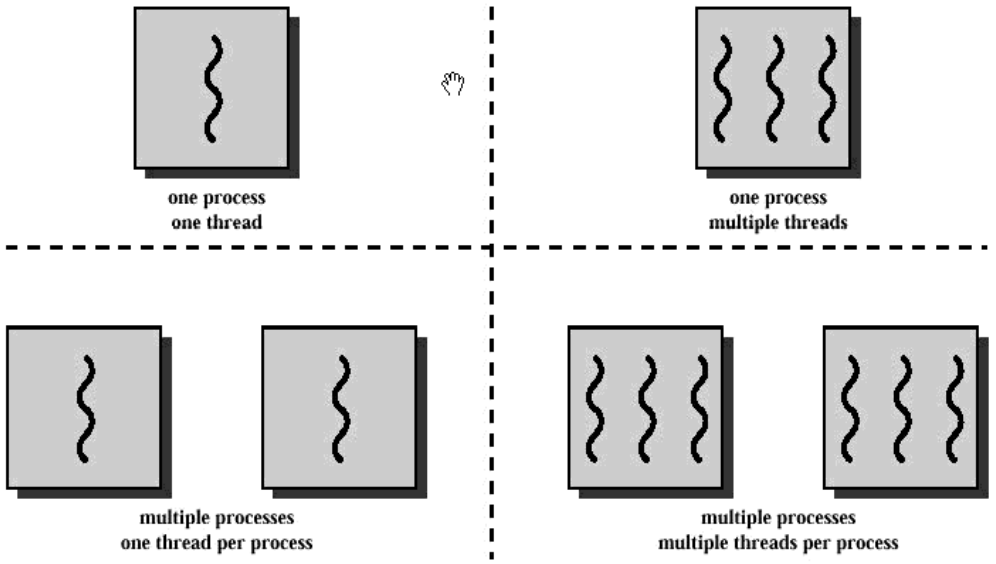
Współbieżność w Javie

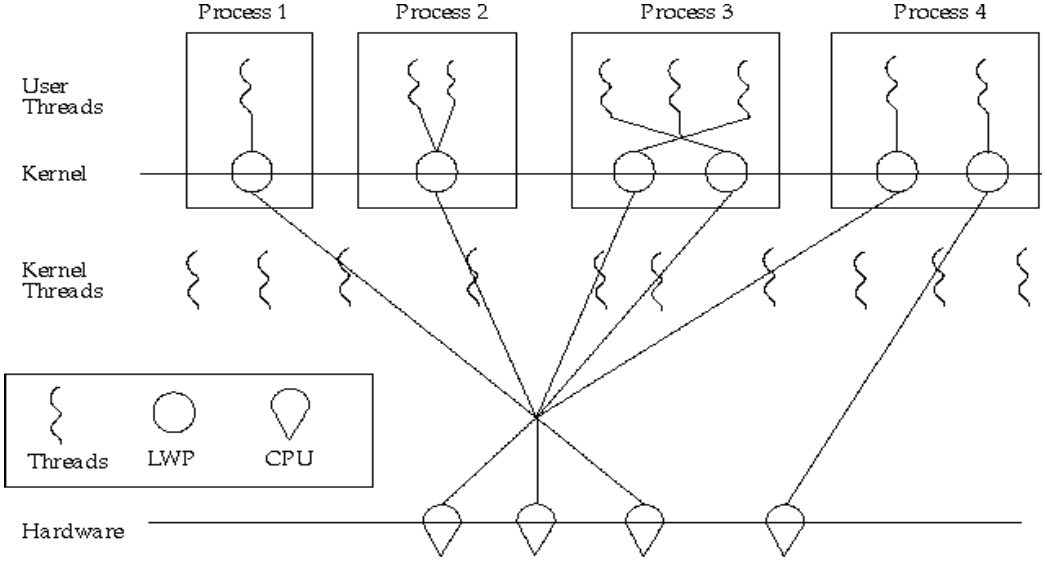
Laboratorium 1

Wątki

1. Wątek a proces



2. Kernel threads, user threads i Leightweight Processes LWP. Mieszany model systemu Solaris



3. standard POSIX threads

Wątki w języku Java

1. Wbudowane w język
2. Dwa rodzaje implementacji
    - Przez dziedziczenie z klasy Thread

Klasa:

```
class MyThread extends Thread {
     . . .
     public void run() {
          . . .
     }
}
```

Tworzenie obiektu i uruchamianie wątku:

```
MyThread t = new MyThread();
t.start();
```

- Przez implementację interfejsu Runnable

Klasa:

```
class MyThreadR implements Runnable {
     . . .
     public void run() {
          . . .
     }
}
```

Tworzenie i uruchamianie:

```
MyThreadR r = new MyThreadR();
Thread t = new Thread(r);
t.start();
```

3. Metoda run()

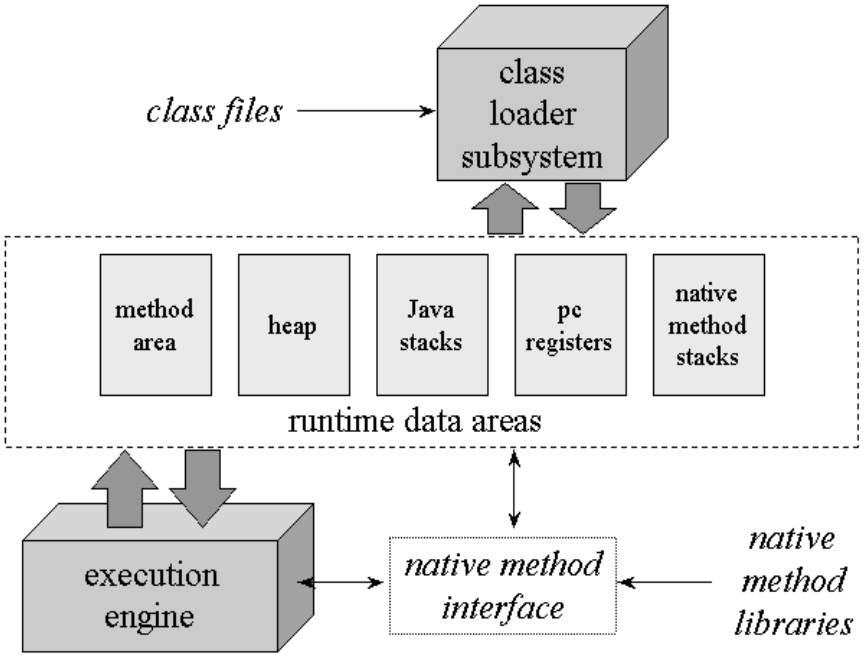
```
4. public void run() {
5.  // kod wątku
6. }
```

7. Klasa Thread
   - Dokumentacja klasy
   - Niektore wazne metody

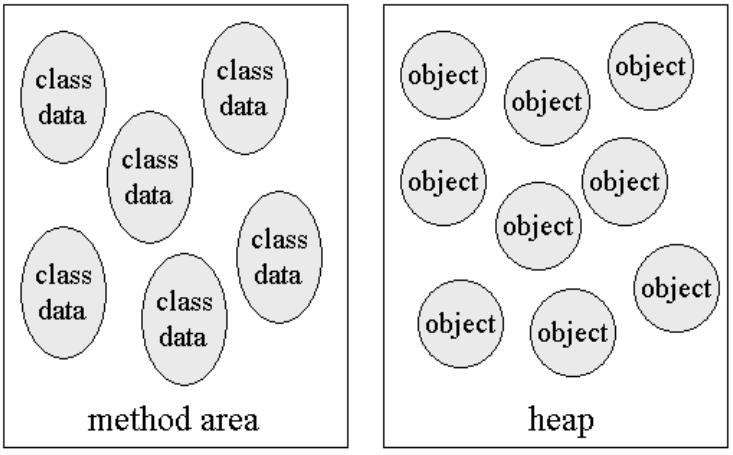
```
o void setName(String name)
o String getName()
o void setPriority(int newPriority)
o int getPriority()
o void join()
o static void sleep(long milis, int nanos)
o public static Thread currentThread()
o String toString()
```

Wątki w JVM

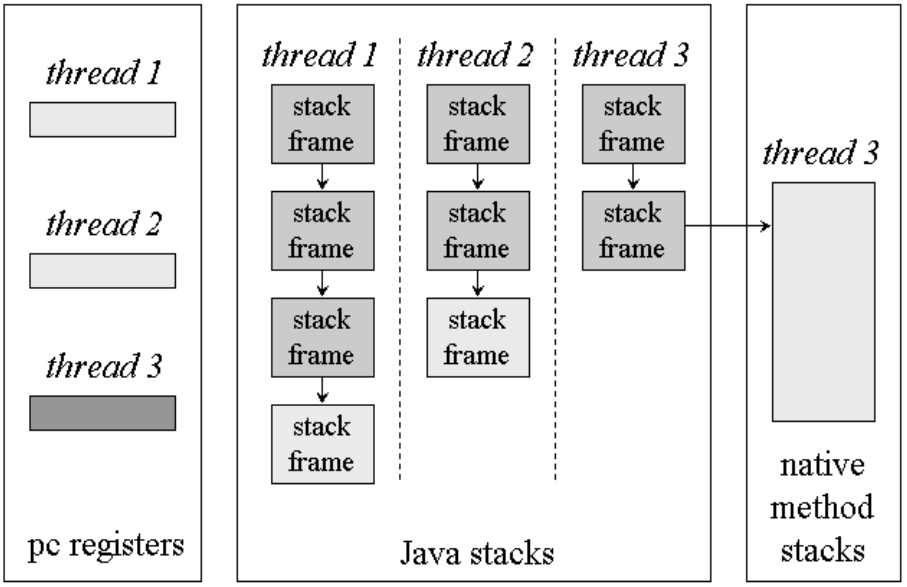
1. Architektura JVM



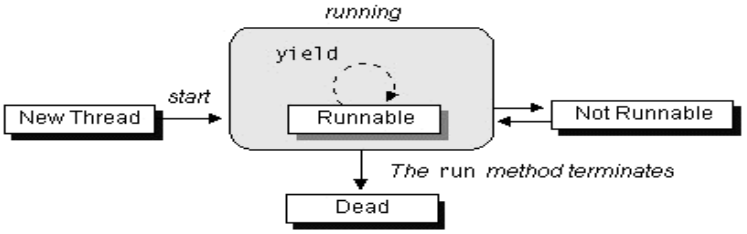
2. Obszary pamięci:
   - Dzielone pomiedzy watki



   - Prywatne dla kazdego watku



3. Cykl życia wątku, stany wątku



4. Szeregowanie wątkow w Javie
    - Ogólne pojęcia szeregowania: wywłaszczania (*preemption*), podziału czasu (*time-slicing*, *time-sharing*), priorytety.
    - Dokładne zachowanie się wątków jest zależne od platformy.
    - Wątki mogą być zaimplementowane całkowicie w przestrzeni użytkownika lub korzystać z natywnych interfejsów platformy.
    - 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.
    - Specyfikacja nie zakłada podziału czasu.
    - 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.
    - Reguly 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)

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)

Ćwiczenie

Zadania.

Do przeczytania

Przeczytać rozdział 20 *Thread Synchronization* z książki Inside the Java Virtual Machine

Zadanie dodatkowe

W systemie dziala **N** wątkow, które dzielą obiekt licznika (początkowy stan licznika = 0).
Każdy wątek wykonuje w pętli **5** razy inkrementację licznika. Zakładamy, że inkrementacja składa się z sekwencji trzech instrukcji: `read`, `inc`, `write` (odczyt z pamięci, zwiększenie o 1, zapis do pamięci). Wątki nie są synchronizowane.
1. Jaka jest teoretycznie najmniejsza wartość licznika po zakończeniu działania wszystkich wątków i jaka kolejność instrukcji (przeplot) do niej prowadzi?
2. Spróbować znaleźć dowód, że będzie to zawsze najmniejsza wartość.

Bibliografia

1. Jacek Rumiński, Język Java. Rozdział o wątkach
2. Bill Venners, Inside the Java Virtual Machine (rozdz. 5, *The Java Virtual Machine*), McGraw-Hill Companies; 2nd Bk&Cdr edition, 2000.

Włodzimierz Funika, funika at agh.edu.pl