Łukasz Oprych gr.5 lab	Programowanie Równoległe	26.11.2023

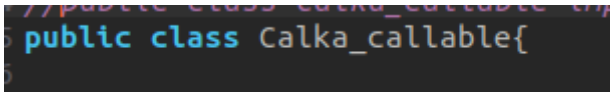
Cel ćwiczenia:

Zapoznanie się z pisaniem programów w języku Java z wykorzystaniem puli wątków i interfejsem Future.

Przebieg ćwiczenia:

Po utworzeniu struktury katalogowej oraz pobraniu plików ze strony prowadzącego, dokonano edycji klasy Calka_callable oraz utworzono sekwencyjną wersję programu.

Sekwencyjna wersja programu:



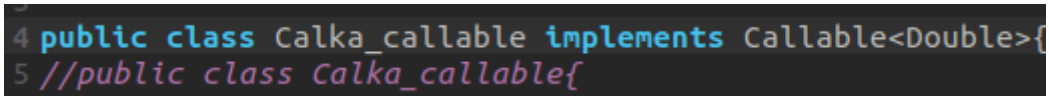

```
public class LiczenieCalkiSekwencyjne {
        public static void main(String[] args) {
                double start = 0;
                double end= Math.PI;
                double dx = 0.000001;
                Calka_callable calka = new Calka_callable(start, end, dx);
                System.out.println("Wynik calki sekwencyjnie: " + calka.compute_integral());
        }
}
```

Wynik programu:

```
Creating an instance of Calka_callable
xp = 0.0, xk = 3.141592653589793, N = 3141593
dx requested = 1.0E-6, dx final = 9.9999988973421 86E-7
Calka czastkowa: 1.9999999999997575
Wynik calki: 1.9999999999997575
loprych@loprych-VirtualBox:~/Desktop/PR_lab/lab_7/lab_Java_threadpool$
```

Wersja równoległa:

Zmiana nagłówka klasy Calka_callable, w celu umożliwienia wykorzystania puli wątków.



Metoda compute_integral() w Calka_callable.

```
public double compute_integral() {
    double calka = 0;
    int i;
    for(i=0; i<N; i++){
        double x1 = xp+i*dx;
        double x2 = x1+dx;
        calka += ((getFunction(x1) + getFunction(x2))/2.)*dx;
    }
    //System.out.println("Calka czastkowa: " + calka);
    return calka;
}
```

Utworzenie metody call() wykonującej metodę compute_integral().

```
    public Double call() throws Exception {
        return compute_integral();
    }
```

Zdefiniowanie zmiennych niezbędnych do obliczeń w wersji równoległej takich jak przedział całkowania (start; end), ilość zadań, ustalenie rozmiaru przedziału na dane zadanie.

```
double start = 0;
double end = Math.PI;
double task = 50;
double dx = 0.000001;
double przedzialPerZadanie = (end - start) / task;
double kX = przedzialPerZadanie;
```

Utworzenie listy obiektów Future<double>, w której będziemy zapisywać obiekty Future.

```
List<Future<Double>> listaWynikow = new ArrayList<Future<Double>>();
```

Utworzenie executora o danej puli wątków.

```
public static void main(String[] args) {
    ExecutorService executor = Executors.newFixedThreadPool(NTHREADS);
```

Utworzenie obiektu klasy Calka_callable, w którym jest przyjmowany przedział całkowania, przedział zadań, oraz dokładności obliczeń w postaci wysokości trapezu.

```
Callable<Double> calkacallable = new Calka_callable(pX, kX, dx);
```

Tworzenie obiektu Future na podstawie stworzonego w tej pętli obiektu Callable i przekazanie go do uprzednio stworzonej listy.

```
for (double pX = start; pX < end; pX += przedzialPerZadanie){
        if(kX > end){kX = end;}
        Callable<Double> calkacallable = new Calka_callable(pX, kX, dx);
        Future<Double> future = executor.submit(calkacallable);
        listaWynikow.add(future);
        kX += przedzialPerZadanie;
}
```

Utworzenie pętli, gdzie następuje iteracja po obiektach Future, w której użyto metody get() zawierającej wyniki zwracane przez metodę call() i sumowane w zmiennej wynik.

```
for (Future<Double> future : listaWynikow){
        try {
        wynik += future.get();
        catch (InterruptedException | ExecutionException e){
                e.printStackTrace();
        }

}
```

Wynik programu:

```
Calka czastkowa: 0.047123116179994984
Calka czastkowa: 0.009827450585788352
Calka czastkowa: 0.049638737456212806
Calka czastkowa: 0.0019732715717281427
Calka czastkowa: 0.06254271403498646
Calka czastkowa: 0.013704089600056294
Calka czastkowa: 0.028520372422153505
Calka czastkowa: 0.01752664483347626
Calka czastkowa: 0.021280030406900304
Finished all threads
Wynik calkowania rownolegle: 1.999999999999835
loprych@loprych-VirtualBox:~/Desktop/PR_lab/lab_7/lab_Java_threadpool$
```

```
loprych@loprych-VirtualBox:~/Desktop/PR_lab/lab_7/lab_Java_threadpool$ java Licz
eniecalki
Creating an instance of Calka_callable
xp = 0.0, xk = 0.06283185307179587, N = 62832
dx requested = 1.0E-6, dx final = 9.999976615704714E-7
Creating an instance of Calka_callable
xp = 0.06283185307179587, xk = 0.12566370614359174, N = 62832
dx requested = 1.0E-6, dx final = 9.999976615704714E-7
Creating an instance of Calka_callable
xp = 0.12566370614359174, xk = 0.1884955592153876, N = 62832
dx requested = 1.0E-6, dx final = 9.999976615704714E-7
Creating an instance of Calka_callable
xp = 0.1884955592153876, xk = 0.25132741228718347, N = 62832
dx requested = 1.0E-6, dx final = 9.999976615704714E-7
```

Wnioski:

Pula wątków to mechanizm służący do implementacji programów równoległych w środowisku Java. Mając do dyspozycji mniej wątków niż zadań, stajemy przed problemem odpowiedniego rozdysponowania zadań. Java udostępnia nam zautomatyzowany proces będący pulą wątku, gdzie nie odwołujemy się bezpośrednio do danego wątku, lecz do puli jako całości i zlecamy puli wykonanie zadania, bez sprawdzenia, który wątek i kiedy to zadanie wykona. Takie rozwiązanie niesie ze sobą dużo zalet, takich jak wygoda programisty czy bezpieczeństwo pamięci. Niestety odejmuje pewnej elastyczności czy sprawia, że program jest mniej przewidywalny dla programisty. Wykorzystanie puli wątku pozwala na sprawne rozwiązanie problemu zrównoleglania większej ilości zadań niż wątków.

Future to mechanizm w środowisku Java pozwalający na przypisanie do obszaru pamięci wyniku, który zostanie obliczony w przyszłości. Dzięki Future mamy możliwość projektowania programów równoległych, gdzie zmienne przechowujące wynik mogą oczekiwać na wykonanie procedury w wątku. Daje to dużą swobodę w implementacji oraz pozwala na bardziej zdefiniowane zarządzanie procesem przebiegu wykonania kodu źródłowego. Obiekt Future pozwala na synchronizację obliczeń równoległych oraz może nam też między innymi mówić o zakończeniu zadania.