Programowanie równoległe. Przetwarzanie równoległe i rozproszone.

Laboratorium 7

Cel:

- nabycie umiejętności pisania programów w języku Java z wykorzystaniem puli wątków

Zajęcia:

1. Utworzenie katalogu roboczego (np. *lab_Java_threadpool*).
2. Napisanie sekwencyjnego programu obliczania całki z zadanej funkcji metodą trapezów korzystając z dostarczonej klasy *Calka_callable* :
   a) do testowania podczas całego laboratorium jak zwykle przykładowo można przyjąć całkowanie ze znanym dokładnym wynikiem, np. całka z *sin(x)*, w przedziale *(0, Math.PI)*
   b) *Calka_callable* oblicza całkę w zadanym przedziale, z granicami przedziału i dokładnością zadaną w konstruktorze (liczba trapezów jest obliczana wewnętrznie przez *Calka_callable* - w tym laboratorium nie należy zajmować się liczbą trapezów!)
      - parametr *dx* (wysokość pojedynczego trapezu w algorytmie całkowania) determinujący dokładność całkowania jest całkowicie niezależnym parametrem zadawanym przez użytkownika (w dalszej części nie zależy np. od liczby zadań)
   c) do obliczenia całki dla zadanych w konstruktorze parametrów służy funkcja *compute_integral()* , którą należy wywołać w wersji sekwencyjnej programu
3. Pobranie paczki java_executor_test.tgz , rozpakowanie, uruchomienie, sprawdzenie poprawności działania.
4. Na podstawie przykładu z java_executor_test.tgz, wykorzystującego interfejs *ExecutorService* oraz klasę *Executors*, zmodyfikowanie programu obliczania całki tak, aby używać puli o stałej liczbie wątków
   a) pojedyncze zadanie dla obiektu typu *Executor* ma stanowić obliczenie całki w podprzedziale, w tym celu zadany przedział (np. (0, PI)) należy podzielić na tyle podprzedziałów, ile jest zadań
      - szerokość podprzedziału dla pojedynczego zadania z założenia jest całkowicie niezależna od wysokości pojedynczego trapezu *dx*, determinującej dokładność całkowania
   b) dla każdego zadania należy utworzyć obiekt klasy *Calka_callable* i przekazać go do puli wątków
      - klasę *Calka_callable* należy zmodyfikować:
        - odkomentować poprawny nagłówek umożliwiający wykorzystanie obiektu klasy jako zadania dla puli wątków
        - uzupełnienie kodu klasy o wymaganą przez interfejs funkcję
   c) liczba wątków i liczba zadań są niezależnymi parametrami (trzecim niezależnym parametrem jest dokładność całkowania, zadawana przez *dx*) - zazwyczaj liczba zadań jest większa od liczby wątków (liczba wątków powinna być związana z liczbą rdzeni procesora, liczba zadań powinna być np. kilka razy większa od liczby wątków, aby umożliwić równoważenie obciążenia)
   d) zadania należy tworzyć i przekazywać do wykonania w jednej pętli
   e) wyniki (korzystając z interfejsu *Future*) powinny być odbierane w kolejnej pętli - tylko tak możliwe jest działanie równoległe
5. Uruchomienie i przetestowanie poprawności działania. **(ocena)**

---------- 3.0 ----------

6. Utworzenie katalogu roboczego (np. *lab_Java_fork_join*).
7. W katalogu roboczym, na podstawie wykładu i dostarczonego szkieletu klasy do wykorzystania w sortowaniu przez scalanie (*scal_tab.java*), utworzenie programu sortowania, wykorzystującego pule wątków Javy, ale tym razem korzystając z klasy *ForkJoinPool*

- uwaga na poprawność wywołania rekurencyjnego - musi kiedyś się kończyć!

8. Uruchomienie i przetestowanie działania. **(ocena)**

---------- 4.0 ----------

Dalsze kroki:

1. Zmodyfikowanie programu obliczania całki, tak aby utworzyć klasę *Calka_runnable*, która implementuje interfejs *Runnable*
   - ze względu na to, że funkcja *run()* nie zwraca wyniku, konieczne jest opracowanie alternatywnego działania
   - np. funkcja *main()* może przesłać do obiektów klasy *Calka_runnable* (w konstruktorze) obiekt, za pomocą którego zostaną zwrócone wyniki (np. obiekt klasy posiadającej synchronizowaną funkcję *dodaj_calke_czastkowa(...)* i funkcje *zwroc_wynik_calkowania()* , którą wywoła funkcja *main()*) lub dowolny inny zaprojektowany mechanizm
   - funkcja *main()* musi:
     - jawnie utworzyć obiekty klasy *Calka_runnable* odpowiednio inicjując je argumentami konstruktora
     - jawnie utworzyć wątki, przesyłając każdemu wątkowi obiekt klasy *Calka_runnable* jako argument konstruktora
2. Zmodyfikowanie programów obliczania histogramu (laboratorium 6), tak aby korzystały z puli wątków (szczególnie dla dekompozycji tablicy znaków, szczególnie dla dekompozycji 2D).

---------- 5.0 ----------

Warunki zaliczenia:

1. Obecność na zajęciach i wykonanie co najmniej kroków 1-5
2. Oddanie standardowego sprawozdania – zgodnie z regulaminem laboratoriów