|  |  |  |
| --- | --- | --- |
| **Łukasz Oprych**  Gr. 5 Lab, 410687 | Programowanie Równoległe | Temat: Java |

**Cel ćwiczenia:**

Celem ćwiczenia było zapoznanie się z tworzeniem programów wielowątkowych oraz opanowanie podstawowych metod synchronizacji w języku Java.

**Przebieg ćwiczenia:**

**Wariant nr 1**

Po pobraniu plików oraz przygotowaniu środowiska zgodnie z poleceniem prowadzącego wykonano pierwszy wariant zadania początkowo poprzez stworzenie nowej klasy Watek w pliku Histogram\_test, która dziedziczy po klasie Thread.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

W klasie dodano konstruktor, gdzie przekazujemy obraz i indeks znaku, jest też wywoływana funkcja run korzystająca z metod klasy Obraz, obliczających ilość powtórzeń danego znaku w histogramie oraz wyświetlanie.

Funkcja main klasy Histogram\_test:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie

Opis wygenerowany automatycznie

Stworzenie w klasie Obraz tablicy histogram\_parallel do przechowywania wyników obliczeń równoległych, początkowo jest wypełniona wartością 0 w metodzie clear\_histogram\_parallel.



Obraz zawierający tekst, Czcionka, zrzut ekranu

Opis wygenerowany automatycznie

Aby obliczyć ilość powtórzonych znaków wykorzystano metodę calculate\_histogram\_parallel, oraz aby sprawdzić czy liczba wystąpień znaku danego wątku jest zgodna w wersji sekwencyjnej oraz równoległej metodę check\_histograms.

Obraz zawierający tekst, zrzut ekranu, Czcionka, oprogramowanie

Opis wygenerowany automatycznie

Stworzono metodę print\_histogram\_parallel do wyświetlania znaku „=” przy powtórzeniu się znaku w danym wątku. Jak widać, użyto synchronized, które odpowiada za zabezpieczenie sekcji krytycznej, aby tylko jeden wątek wykonywał ten fragment kodu do czasu opuszczenia przez niego tej sekcji.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Stworzenie i uruchomienie wątków dla pierwszego przypadku:

Obraz zawierający tekst, Czcionka, zrzut ekranu, linia

Opis wygenerowany automatycznie

Wynik dla tablicy o wymiarze 10x10:

Obraz zawierający zrzut ekranu, Czcionka, tekst

Opis wygenerowany automatycznieObraz zawierający tekst, zrzut ekranu, Czcionka, design

Opis wygenerowany automatycznie

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

**Wariant nr 2**

W tym przypadku dochodzi do dekompozycji blokowej.

Utworzenie klasy wątek, która implementuje interfejs Runnable, oraz zmodyfikowanie jej pod dany wariant.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Modyfikacja funkcji main:

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Zmienna divisionRest gwarantuje, że póki będzie wychodzić więcej niż 0, to te dane będą przekazywane do pierwszych wątków. (Żeby ostatni watek nie miał dużo danych, w celu zapewnienia wydajności).

Utworzenie funkcji calculate\_histogram\_parallel, oraz funkcji print\_histogram\_parallel.

Obraz zawierający tekst, zrzut ekranu

Opis wygenerowany automatycznie

Wynik dla tablicy 20x20 oraz zastosowanych czterech wątków:

Obraz zawierający tekst, zrzut ekranu, Czcionka, design

Opis wygenerowany automatycznieObraz zawierający tekst, zrzut ekranu, Czcionka, projekt graficzny

Opis wygenerowany automatycznie

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

**Wnioski:**

W Javie istnieją dwa główne podejścia do tworzenia nowych wątków:

- Poprzez utworzenie własnej klasy, która dziedziczy po klasie Thread. Możemy stworzyć obiekt tej klasy i rozpocząć wątek. Użycie podklasy Thread zajmuje więcej pamięci i nie pozwala na rozszerzanie innej klasy, ponieważ rozszerza już Thread. Pozwala jednak startować bez tworzenia nowej instancji Thread.

- Poprzez utworzenie klasy implementującej interfejs Runnable. Aby uruchomić wątek, musimy przekazać instancję tej klasy jako argument do konstruktora nowego obiektu klasy Thread. Implementacja interfejsu Runnable zajmuje mniej pamięci i daje więcej możliwości, pozwala aby nasza klasa rozszerzała jeszcze inne klasy. Do uruchomienia wątku potrzebuje jednak instancji Thread.

Przy tworzeniu programu, trzeba rozważyć czy dla naszego programu bardziej ważne jest rozszerzanie innych klas czy prostota uruchamiania wątków.

Oznaczenie metody jako synchronized pozwala na zapewnienie bezpieczeństwa działania obiektu pod kątem spójności operacji w przypadku wielowątkowym.