**Sprawozdanie: Implementacja histogramu znaków ASCII w Javie**

**Opis zadania:**

Celem zadania było obliczenie histogramu częstotliwości występowania znaków ASCII w dwuwymiarowej tablicy znaków.

Program zaimplementowano w dwóch wersjach: **sekwencyjnej** (pojedynczy wątek) oraz **równoległej** (wiele wątków). W wersji równoległej wyróżniono dwa warianty: (a) *wariant 1* – każdy wątek liczy wystąpienia dokładnie jednego znaku ASCII (klasa rozszerzająca Thread), (b) *wariant 2* – każdy wątek odpowiada za blok kilku kolejnych znaków ASCII (klasa implementująca Runnable). Program najpierw generuje tablicę o zadanych wymiarach wypełnioną losowymi znakami ASCII, następnie oblicza histogram sekwencyjnie i wypisuje wynik, a potem uruchamia obliczenia równoległe (tworzy odpowiednie wątki, scala wyniki) i porównuje je z wersją sekwencyjną.

Zrealizowane kroki:

* Generacja danych – utworzenie tablicy n×m z losowymi znakami ASCII i wyświetlenie jej.
* Obliczenie histogramu sekwencyjnie (jednowątkowo): zliczenie wystąpień każdego znaku.
* *Wariant 1 (Thread per char)* – uruchomienie 94 wątków, każdy liczy wystąpienia przypisanego znaku.
* *Wariant 2 (Runnable, blokowy)* – podział 94 znaków na zadaną liczbą wątków liczbę bloków; każdy wątek liczy znaki ze swojego bloku.
* Porównanie otrzymanych histogramów i wyświetlenie informacji o zgodności wyników.

**Kod:**

****

**Wnioski**

* **Poprawność:** Obie wersje programu powinny dawać identyczne wyniki histogramu (co weryfikujemy porównując tablice wyników). Testy pokazały, że połączenie wyników wątków (zarówno w wariancie 1 jak i 2) zgadza się z obliczeniem sekwencyjnym. Program informuje o *zgodności histogramów* po zakończeniu obliczeń.
* **Wydajność:** Wersja równoległa może być szybsza przy dużych danych i gdy liczba wątków jest dostosowana do liczby rdzeni procesora. Należy jednak pamiętać, że tworzenie i zarządzanie wątkami generuje narzut czasowy. Uruchomienie bardzo wielu wątków (np. 94 wątki dla każdego znaku) może być mniej efektywne od użycia mniejszej liczby wątków, z powodu przełączania kontekstu i kosztów administracyjnych.
* **Zalecenia konstrukcyjne:** W Javie zwykle korzystniej jest implementować Runnable niż rozszerzać klasę Thread, ponieważ pozwala to oddzielić logikę zadania od mechanizmu wykonania. Dzięki temu kod jest bardziej elastyczny (np. można wykorzystać pule wątków). W zadaniu pokazano obie metody (klasę Thread i Runnable) zgodnie z wymaganiami ćwiczenia.
* **Zrzuty ekranu:**



