Praca domowa 05 – prod

Michał Szczygieł

Wstęp

Zadanie składa się z 2 plików:

- Main.java
- Prod.java

CEL: Odczytanie danych z pliku źródłowego przez producenta i przekazani wartości w postaci kolejki do konsumentów. Wszystkie wątki pracują w czasie zgodnym w zadanym rozkładzie normalnym. W końcowym etapie wyświetlenie różnicy wartości średnich ‘obsłużonych’ przez konsumentów liczb .

Klasa Main

Zadaniem tej klasy jest wczytanie danych z pliku, oraz przekazanie tych wartości do klasy Prod. Również oprócz danych z pliku klasa ta winna przekazać parametr size, który również jak i parametr file path nadawana jest jako argument wywoławczy programu.

Metody:

```
public static void main(final String... args) throws FileNotFoundException
```

- Metoda główna, przyjmuje dwa parametry filePath i size, przekazuje wczytane dane i rozmiar do klasy Prod, wykonując cel, oraz wyświetlenie wyniku z dokładnością do trzech miejsc po przecinku.

```
private static LinkedList<Integer> readFile(File file) throws FileNotFoundException
```

– Metoda ta wczytuje dane w postaci liczb całkowitych z podanej ścieżki oraz zwraca przefiltrowane dane.

Klasa Prod

Klasa ta, otrzymawszy dane od klasy wywołującej, ustawia te dane w konstruktorze oraz uruchamia trzy wątki w postaci dwóch konsumentów i jednego producenta. Klasa ów zawiera także 4 klasy wewnętrzne:

```
private class Queue
private abstract class Worker extends Thread
private class Producer extends Worker
private class Consumer extends Worker
```

Powyższe klasy wewnętrzne zostały opisane w kodzie źródłowym programu.

Realizacja algorytmu Prod

W mojej realizacji algorytmu dla klasy Prod zastosowałem następujące kroki.

Tworzę klasę Queue, która jest pewnego rodzaju kontenerem, gdyż przechowuje ona dane z pliku wejściowego jak i zawiera kolejkę danych obsługiwaną przez producenta i konsumentów. W niej odbywają się operacje wstawiania i pobierania elementów z kolejki.

W kolejnym etapie tworzone są wątki producenta i konsumentów którym przekazywane zostają następujące rzeczy takie jak:

- wartość m
- wartość sigma
- obiekt Queue
- Dla konsumentów zostaje przekazana dodatkowo wartość o granicy pobierania danych z kolejki.

Wątek producenta wstawia dane do kolejki (o ile to możliwe, gdyż kolejka ma określony rozmiar size) pobierając dane z pliku wejściowego robiąc to w losowych odstępach czasu zadanego z rozkładu normalnego.

```
public static long normalDistribution(Double mean, Double sigma) {
  return (long) Math.abs(1000 * (rand.nextGaussian() * sigma + mean));
}
```

Wątki konsumentów pobierają dane z kolejki robiąc to w losowych odstępach czasu zadanego z rozkładu normalnego. Działanie ich nie kończy się wtedy kiedy wczytają wszystkie dane z kolejki, bo czekają również na zakończenie działania wątku producenta.

Po skończeniu operacji klasa Prod zwraca wynik w postaci różnicy średnich wartości obsłużonych liczb przez wątki. Złożoność obliczeniowa dla tego programu wynosi:

w optymistycznym przypadku $O(n)$ a w pesymistycznym $O(n^2)$.

Dyskusja

W przypadku realizacji rozwiązania tego programu, wydaje się że nie istnieją inne rozwiązania, gdyż ograniczenia zadane w treści zadnia wyraźnie określają w jaki sposób ma odbywać się program. Różnica jedynie może wynikać w postaci dobory struktury programu, oraz mechanizmów rozkładu pracy. W moim wypadku zamiast używać metody sleep(), co moim zdaniem nie wygląda ładnie, można było użyć TimerTask'ów.