Projekt programistyczny z przedmiotu SKJ – semestr zimowy 2022/23

Temat: **Rozproszona baza danych**

Opis:
Na rozproszoną bazę danych składa się zbiór procesów rozmieszczonych w sieci. Każdy z procesów posiada pewien zbiór informacji (dla uproszczenia założymy, że będzie to jedna para <klucz, wartość> na jeden proces). Procesy tworzą sieć, w której dany proces jest połączony z co najmniej jednym innym procesem z sieci. Sieć jest tworzona w sposób przyrostowy poprzez uruchamianie kolejnych procesów (tworzenie kolejnych węzłów) i podłączanie ich do już istniejącej sieci. Każdy proces (poza pierwszym) po starcie podpina się do sieci zgodnie z założonym modelem komunikacji w sieci i oczekuje na ewentualne połączenia od kolejnych nowych składowych systemu (kolejnych dodawanych węzłów) lub klientów pragnących wykonać operacje na bazie. Komunikacja z potencjalnymi klientami oraz kolejnymi nowo podłączanymi węzłami odbywa się za pomocą protokołu TCP. Komunikacja z węzłami tworzącymi sieć w trakcie wykonywania operacji na bazie może odbywać się zarówno za pomocą TCP jak i UDP – wybór pozostaje w gestii realizatora zadania. Wszystkie procesy są symetryczne czyli zarówno przechowują swoje dane jak i przyjmują zlecenia operacji od klientów. Węzły komunikują się wyłącznie z węzłami, do których same podłączyły się przy starcie lub tymi, które połączyły się z nimi.

Uruchomienie węzła sieci ma następującą postać:

```
java DatabaseNode -tcpport <numer portu TCP> -record <klucz>:<wartość>
     [ -connect <adres>:<port> ]
```

gdzie:

- `-tcpport <numer portu TCP>` określa numer portu TCP na którym dany węzeł sieci oczekuje na połączenia od klientów.
- `-record <klucz>:<wartość>` oznacza parę liczb całkowitych początkowo przechowywanych w bazie na danym węźle, gdzie pierwsza to klucz a druga to wartość związana z tym kluczem. Nie ma wymogu unikalności zarówno klucza jak i wartości.
- `[ -connect <adres>:<port> ]` oznacza listę innych węzłów już będących w sieci, z którymi dany węzeł ma się połączyć i z którymi może się komunikować w celu wykonywania operacji. Lista może zawierać wiele węzłów. Dla pierwszego węzła w sieci ta lista jest pusta.

Przykład wywołania:

```
java DatabaseNode -tcpport 9991 -record 17:256 -connect localhost:9990
     -connect localhost:9997 -connect localhost:9989
```

Oznacza to uruchomienie węzła, który pod kluczem o wartości 17 przechowuje wartość 256, nasłuchuje na połączenia od klientów lub innych nowych węzłów na porcie 9991 a do podłączenia siebie do sieci wykorzysta węzły pracujące na komputerze o adresie `localhost` i portach TCP o numerach 9990, 9997 i 9989.

Po uruchomieniu sieci, można do niej podpinać klientów. Klient jest prostym procesem, który ma za zadanie wysłać do sieci żądanie wykonania operacji w odpowiednim formacie i poczekać na odpowiedź. Podczas uruchomienia klient otrzymuje jako parametr adres IP jednego z węzłów sieci (dowolnego, zwanego dalej „węzłem kontaktowym”), numer portu TCP na którym ten węzeł nasłuchuje na zgłoszenia od klientów oraz operację do wysłania do węzła (w formie tekstowej). Uruchomienie klienta odbywa się w następujący sposób:

```
java DatabaseClient -gateway <adres>:<numer portu TCP>
     -operation <operacja z parametrami>
```

Przykładowe wywołanie klienta:

```
java DatabaseClient -gateway localhost:9991 -operation get-value 17
```

oznacza uruchomienie klienta, który do połączenia z siecią ma wykorzystać węzeł działający na komputerze `localhost` i porcie TCP o numerze 9991. Po połączeniu baza ma podjąć próbę znalezienia wartości przypisanej do klucza 17.

Po uzyskaniu połączenia klient przesyła w formie pojedynczej linii tekstu komunikat o następującym formacie:

```
<operacja> [ <parametr> ]
```

(będącą tekstem po parametrze `-operation`) gdzie poszczególne pola zależą od wykonywanej operacji. Dostępne operacje wraz z parametrami to:

1. `set-value <klucz>:<wartość>` : ustawienie nowej wartości (drugi parametr) dla klucza będącego pierwszym parametrem. Wynikiem operacji jest komunikat `OK` jeśli operacja się powiodła lub `ERROR` jeśli baza nie zawiera żadnej pary, w której występuje żądany klucz. Jeśli w bazie jest kilka rekordów o takim samym kluczu, zmianie musi ulec co najmniej jeden z nich.
2. `get-value <klucz>` : pobranie wartości dla klucza będącego parametrem w bazie. Wynikiem operacji jest komunikat składający się z pary `<klucz>:<wartość>` jeśli operacja się powiodła lub `ERROR` jeśli baza nie zawiera żadnej pary, w której występuje żądany klucz. Jeśli baza zawiera więcej niż jedną parę z takim kluczem, tylko jeden wynik musi zostać zwrócony (którykolwiek).
3. `find-key <klucz>` : zlecenie wyszukania adresu i numeru portu węzła, na którym przechowywany jest rekord o zadanym kluczu. Jeśli taki węzeł istnieje, odpowiedzią jest para postaci `<adres>:<port>` identyfikująca ten węzeł lub komunikat `ERROR` jeśli żaden węzeł takiego klucza nie posiada. Jeśli baza zawiera więcej niż jedną parę z takim kluczem, tylko jeden wynik musi zostać zwrócony (którykolwiek).
4. `get-max` : znalezienie największej wartości przypisanej do wszystkich kluczy w bazie. Wynikiem operacji jest komunikat składający się z pary `<klucz>:<wartość>`. Jeśli baza zawiera więcej niż jedną parę z takim kluczem, tylko jeden wynik musi zostać zwrócony (którykolwiek).
5. `get-min` : znalezienie najmniejszej wartości przypisanej do wszystkich kluczy w bazie. Wynikiem operacji jest komunikat składający się z pary `<klucz>:<wartość>`. Jeśli baza zawiera więcej niż jedną parę z takim kluczem, tylko jeden wynik musi zostać zwrócony (którykolwiek).
6. `new-record <klucz>:<wartość>` : zapamiętanie nowej pary klucz:wartość w miejsce pary przechowywanej na węźle, do którego dany klient jest podłączony. Wynikiem tej operacji jest komunikat `OK`.
7. `terminate` : powoduje odłączenie się węzła od sieci poprzez poinformowanie o tym fakcie swoich sąsiadów oraz zakończenie pracy. Sąsiedzi węzła poinformowani o zakończeniu przez niego pracy uwzględniają ten fakt w swoich zasobach i przestają się z nim komunikować. Przed samym zakończeniem pracy węzeł odsyła do klienta komunikat `OK`.

Na każde żądanie składa się dokładnie jedna linia w formacie jak powyżej zakończona znakiem nowej linii (uwaga na możliwe różne kodowania). Węzeł systemu po odebraniu takiego komunikatu podejmuje próbę wykonania operacji. W szczególności operacje 1-5 wymagają komunikacji sieciowej w celu odnalezienia węzła odpowiedzialnego za wymagane rekordy, operacja 6 wymaga jedynie operacji w obrębie węzła, z którym klient się komunikuje a operacja 7 komunikacji z sąsiadami węzła. Sposób realizacji operacji oraz schemat komunikacji pozostaje w gestii realizatora zadania – zabrania się jednak użycia modelu z centralnym serwerem posiadającym wiedzę o wszystkich rekordach w sieci. Wynikiem operacji jest jedna linia tekstu zależna od operacji i jej wyniku zakończona znakiem końca linii. Po otrzymaniu odpowiedzi klient wyświetla ją na ekranie i kończy pracę.

Uwagi:

- Na danym fizycznym komputerze może być uruchomionych wiele węzłów bazy danych. Należy więc odpowiednio dbać o wykorzystanie portów aby uniknąć konfliktów.
- Węzły uczestniczące w operacjach mogą wypisywać na konsoli komunikaty informujące o ich przebiegu, np. kto o co pyta lub czego żąda.
- Jeśli żądanie zmiany wartości nie może zostać spełnione, po zakończeniu próby jego wykonania system ma znajdować się w takim stanie jak przed próbą.
- Należy koniecznie przestrzegać zarówno nazewnictwa głównej klasy (jak w przykładach powyżej) jak i sposobu uruchamiania aplikacji – do testów zostanie użyty zautomatyzowany system nietolerujący odchyleń.
- Nie trzeba pisać procesu klienta – dostarczymy go Państwu w formie gotowej.

Specyfikacja:

1 Zadanie polega na zaprojektowaniu i zaimplementowaniu protokołu oraz aplikacji go wykorzystującej, która umożliwi organizację rozproszonej bazy danych oraz późniejsze wykonywanie na niej operacji przez procesy klienckie – zgodnie z opisem. Określenie sposobu organizacji sieci węzłów, podłączania nowych do już istniejącej sieci, oraz metody ich komunikacji w celu realizacji operacji jest elementem zadania.

2 Projekty powinny zostać zapisane do odpowiednich katalogów (w sekcji Zadania) w systemie GAKKO do 15.01.2023 (termin może zostać zmieniony przez prowadzącego daną grupę).

3 Za poprawne rozwiązanie tego zadania można uzyskać do 500 punktów:

3.1 Maksymalnie 200 punktów za kompletną dokumentację opisującą sposób organizacji sieci, opis komunikacji wraz z opisem przesyłanych komunikatów i protokołu komunikacji. **Obecność poprawnej dokumentacji jest warunkiem koniecznym do tego, aby zostały sprawdzone pozostałe elementy projektu** (3.2, 3.3 i 3.4)**.**

3.2 Maksymalnie 100 punktów za implementację procesu węzła sieci umożliwiającą organizację takiej sieci (podłączanie kolejnych węzłów oraz terminację całości).

3.3 Maksymalnie kolejne 100 punktów za uzupełnienie procesu o funkcjonalności pozwalające na odbieranie żądań od klienta i wykonanie operacji przy założeniu, że w danym momencie tylko jeden klient operuje na bazie (konieczna implementacja funkcjonalności punktu 3.2).

3.4 Maksymalnie kolejne 100 punktów za implementację systemu umożliwiającą pracę przy większej niż 1 liczbie klientów korzystających w danym momencie z bazy (konieczna implementacja funkcjonalności punktów 3.2 i 3.3).

4 Spakowane archiwum z plikami projektu musi zawierać:

4.1 Pliki źródłowe (dla JDK 1.8),

4.2 Pliki binarne (class),

4.3 Skrypty startowe, umożliwiające uruchomienie implementacji (opcjonalne),

4.4 Plik Readme.txt z opisem i uwagami autora, w szczególności:

4.4.1 **szczegółowy opis implementacji**, w szczególności szczegółowy opis protokołu (opis budowy komunikatów i ich przetwarzania, wymagane dla punktu 3.4).

4.4.2 jak skompilować i zainstalować (z linii poleceń bez używania GUI),

4.4.3 co zostało zaimplementowane,

4.4.4 co nie działa (jeśli nie działa).

5 JEŚLI NIE WYSZCZEGÓLNIONO INACZEJ, WSZYSTKIE NIEJASNOŚCI NALEŻY PRZEDYSKUTOWAĆ Z PROWADZĄCYM ZAJĘCIA POD GROŹBĄ NIEZALICZENIA PROGRAMU W PRZYPADKU ICH NIEWŁAŚCIWEJ INTERPRETACJI.