

Dokumentacja projektu – Zadanie TCP+UDP: Scentralizowany System Obliczeniowy

Autor: Aleksandra Fus

Numer indeksu: s30395

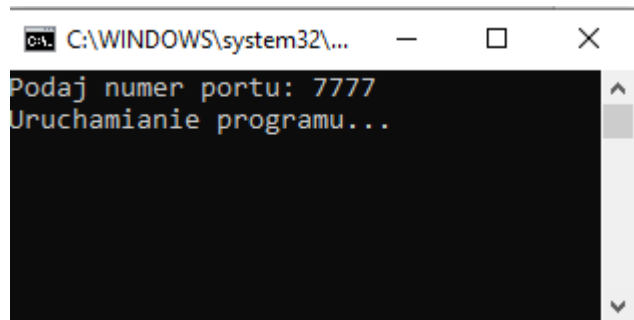
Uruchomienie programu

Za kompilację oraz uruchomienie programu, odpowiedzialny jest plik **runCCS.bat**.

Po jego uruchomieniu, należy podać numer portu, na którym będzie działał program.

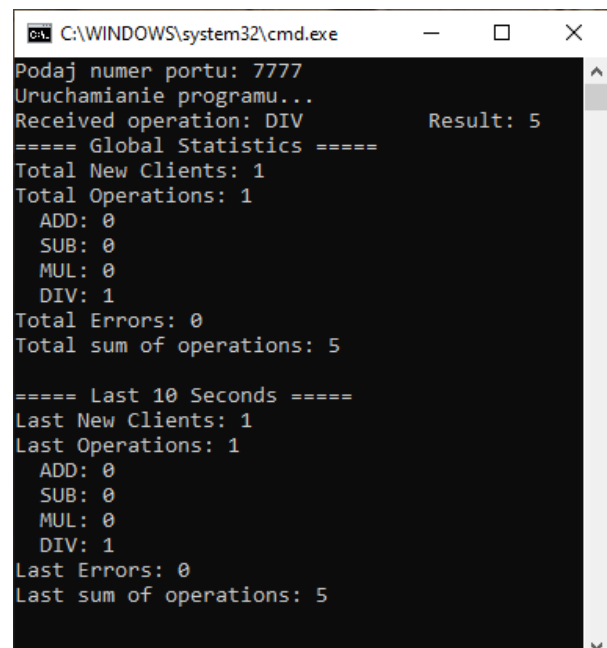
Przykład działania pliku runCCS.bat:

Pierwsze uruchomienie programu.



```
C:\WINDOWS\system32\...  
Podaj numer portu: 7777  
Uruchamianie programu...
```

Program po otrzymaniu komunikatu od klienta oraz przedstawiający statystyki.



```
C:\WINDOWS\system32\cmd.exe  
Podaj numer portu: 7777  
Uruchamianie programu...  
Received operation: DIV Result: 5  
==== Global Statistics ====  
Total New Clients: 1  
Total Operations: 1  
  ADD: 0  
  SUB: 0  
  MUL: 0  
  DIV: 1  
Total Errors: 0  
Total sum of operations: 5  
  
==== Last 10 Seconds ====  
Last New Clients: 1  
Last Operations: 1  
  ADD: 0  
  SUB: 0  
  MUL: 0  
  DIV: 1  
Last Errors: 0  
Last sum of operations: 5
```

Opis projektu

WERSJA JAVA:

Java 8

KLASA CCS

Jest to główna klasa programu, która inicjalizuje jego działanie. W konstruktorze przyjmuje jeden parametr typu int, który jest portem, na którym działają serwisy UDP oraz TCP. Zaczyna również wykonywanie trzech równoległych wątków:

- UDPservice() – odpowiedzialny za komunikacje UDP, pozwalającą klientom na wykrycie serwisu,
- TCPservice() – odpowiedzialny za komunikacje TCP tzn. przyjmowanie komunikatów z działaniami oraz obsługę klientów,
- StatsReporter() – odpowiedzialny za raportowanie statystyk co 10 sekund.

KLASA UDPSERVICE

Implementuje interfejs Runnable oraz metodę run(). W bloku try-catch, na porcie podanym wcześniej przez użytkownika tworzone jest gniazdo DatagramSocket. Wywoływana jest na nim metoda setBroadcast(true), która aktywuje flagę SO_BROADCAST i umożliwia rozsyłanie oraz otrzymywanie datagramów rozgłoszeniowych. Dzięki temu, aplikacja jest w stanie obsłużyć klienta, który szuka serwisu działającego w sieci lokalnej, poprzez wysłanie pakietu rozgłoszeniowego o wymaganej treści.

Następnie, dopóki wątek nie zostanie przerwany, wykonywana jest główna logika serwisu UDP. Tworzona jest tablica bajtów o rozmiarze 12, ponieważ wymagana wiadomość „CCS DISCOVER” posiada dokładnie taki rozmiar. Tworzona jest instancja DatagramPacket packet, przyjmująca jako argumenty tablicę bajtów oraz jej długość. Na początkowo zadeklarowanym gnieździe, wywoływana jest metoda blokująca receive(packet), która czeka na otrzymanie od klienta datagramu. Po otrzymaniu komunikatu, sprawdzane jest, czy jego treść zaczyna się od „CCS DISCOVER”, co instruuje program do odesłania pakietu z treścią „CCS FOUND”. Odbywa się to poprzez stworzenie nowego obiektu typu DatagramPacket sendingPacket z argumentami: tablica z wiadomością przekonwertowaną na ciąg bajtów, długość tej tablicy, adresem wcześniej odebranego pakietu, portem wcześniej odebranego pakietu. Obiekt sendingPacket zostaje wysłany do klienta poprzez początkowo stworzone gniazdo. Wszelkie wyjątki należące do klasy IOException, bądź dziedziczące po niej, są przechwytywane i obsługiwane.

KLASA TCPSERVICE

Implementuje interfejs Runnable oraz nadpisuje metodę run(). Jedynym prywatnym polem klasy jest ExecutorService threadPool, a konkretnie CachedThreadPool(). Zawiera on i zarządza wszystkimi wątkami klientów TCP, tak, aby ich obsługa była realizowana równolegle.

Metoda run() zawiera blok try-catch, w którym tworzone jest gniazdo typu ServerSocket na porcie ustalonym podczas uruchamiania programu. Następnie, dopóki wątek nie zostanie przerwany, tworzone jest gniazdo typu Socket, do którego przypisywane jest gniazdo otrzymane, jako wynik metody blokującej, wywołanej na instancji ServerSocket accept(). W tym momencie nawiązane zostaje połączenie TCP (poprzez 3 way handshake). Do obiektu threadPool dodawany jest nowy wątek TCPClientHandler, przyjmujący jako argument wcześniej zadeklarowany socket klienta. Statystyki programu są aktualizowane.

Wszelkie wyjątki należące do klasy IOException, bądź dziedziczące po niej, są przechwytywane i obsługiwane, a w bloku finally gwarantowane jest zakończenie wykonywania wątków klientów.

KLASA TCPCLIENTHANDLER

Implementuje interfejs Runnable, a w konstruktorze przyjmuje obiekt typu Socket socket. Jedynym polem klasy jest obiekt typu TCPHandler handler, który implementuje główną logikę protokołu TCP.

W metodzie run() odczytywana jest otrzymana wiadomość oraz zależnie od jej zawartości wykonywane są odpowiednie operacje arytmetyczne. W przypadku wykrycia błędu, wywoływana jest metoda error() oraz zakończenie obsługi klienta. Na konsole wypisywane są odpowiednie komunikaty, a statystyki są aktualizowane. Poprzez obiekt TCPHandler wysyłana jest wiadomość do klienta, zawierająca wynik działania programu, bądź informację o błędzie.

KLASA TCPHANDLER

Implementuje główną logikę protokołu TCP. W konstruktorze przyjmuje obiekt typu Socket socket.

W metodzie getMessage() tworzona jest tablica bajtów o rozmiarze 1, gdyż następnie w bloku try-catch do obiektu typu StringBuilder będzie doklejany jeden znak (char o rozmiarze 1 bajt), odczytywany przez InputStream. Obiekt typu InputStream jest wynikiem metody getInputStream() wywołanej na wcześniej zadeklarowanym obiekcie socket. Cały proces odczytywania znaków i łączenia ich trwa, dopóki nie zostaną przeczytane wszystkie dane ze strumienia lub odczytany znak jest znakiem nowej linii.

Metoda sendMessage(String message) służy do wysłania wiadomości do klienta.

Odbywa się to poprzez stworzenie tablicy bajtów, zawierającej przekonwertowaną

wiadomość na ciąg bajtów oraz stworzenie obiektu typu `OutputStream`, otrzymanego w wyniku wywołania metody `getOutputStream()` na wcześniej zadeklarowanym obiekcie `socket`. Wywołując na nim metodę `write(tablica bajtów)`, do strumienia zapisywana jest zawartość tablicy. Ostatecznie, dzięki metodzie `flush()` wywołanej również na obiekcie `OutputStream`, wymuszane jest zapisanie wszystkich danych bufora do strumienia. Metoda `close()` zamyka `Socket socket` i przechwytuje ewentualne wyjątki.

KLASA `STATSREPORTER`

Implementuje interfejs `Runnable` i nadpisuje metodę `run()`, w której usypia wątek na 10 sekund, wypisuje na konsoli statystyki globalne tzn. od początku działania aplikacji i uruchomienia wątku oraz statystyki z ostatnich 10 sekund. Zawiera zmienne `AtomicInteger`, dzięki czemu zapewnia bezpieczny dostęp do danych w środowisku wielowątkowym. Zmienne te są używane do przechowywania liczby klientów, operacji, błędów oraz sumy wyników operacji.