**Zadanie TCP+UDP: Scentralizowany System Obliczeniowy SKJ (2024)**

[Wstęp 2](#_Toc186992633)

[Budowa 2](#_Toc186992634)

[Klasa CCS 2](#_Toc186992635)

[Klasa UDPHandler 2](#_Toc186992636)

[Klasa UDPServer 3](#_Toc186992637)

[Klasa UDPTaskHandler 3](#_Toc186992638)

[Klasa TCPHandler 4](#_Toc186992639)

[Klasa TCPServer 5](#_Toc186992640)

[Klasa TCPStatisticsHandler 5](#_Toc186992641)

[Klasa TCPTaskHandler 7](#_Toc186992642)

[Logika 8](#_Toc186992643)

[Serwer UDP 8](#_Toc186992644)

[Serwer TCP 8](#_Toc186992645)

[Ograniczenia 8](#_Toc186992646)

Wstęp

Dokumentacja zawiera opis budowy oraz logiki aplikacji realizującej Scentralizowany System Obliczeniowy.

Budowa

Klasa CCS

Aplikacja główna uruchamiająca serwery UDP i TCP oraz obsługująca port.

* **Metody:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nazwa** | **Typ zwracany** | **Parametry** | **Opis** |
| main(String[] args) | void | String[] args - Tablica argumentów wejściowych. Oczekuje jednej liczby całkowitej reprezentującej numer portu. | Metoda uruchamiająca aplikację. Parsuje argumenty (port) i uruchamia dwa serwery w osobnych wątkach: serwer UDP i serwer TCP. |
| getPort(String[] args) | int | String[] args - Tablica argumentów wejściowych. | Pobiera port z argumentów wejściowych. Sprawdza, czy port jest liczbą całkowitą w przedziale 1-65535. |

Klasa UDPHandler

Klasa obsługująca operacje wysyłania wiadomości w protokole UDP.

* **Pola:**

|  |  |  |
| --- | --- | --- |
| **Nazwa** | **Typ** | **Opis** |
| socket | DatagramSocket | Gniazdo sieciowe używane do wysyłania pakietów UDP. |

* **Metody:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nazwa** | **Typ zwracany** | **Parametry** | **Opis** |
| sendMessage(InetAddress address, int port, byte[] message) | void | InetAddress address – adres docelowy.  int port – numer portu docelowego.  byte[] message – wiadomość do wysłania. | Wysyła wiadomość do określonego adresu i portu. |

Klasa UDPServer

Serwer UDP, który odbiera wiadomości od klientów.

* **Pola:**

|  |  |  |
| --- | --- | --- |
| **Nazwa** | **Typ** | **Opis** |
| socket | DatagramSocket | Gniazdo sieciowe używane do wysyłania pakietów UDP. |

* **Metody:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nazwa** | **Typ zwracany** | **Parametry** | **Opis** |
| run() | void | - | Metoda uruchamiająca serwer w wątku. Odbiera pakiety i przekazuje je do obsługi. |

Klasa UDPTaskHandler

Klasa obsługująca zadania, która przetwarza odebrane pakiety UDP.

* **Pola:**

|  |  |  |
| --- | --- | --- |
| **Nazwa** | **Typ** | **Opis** |
| handler | UDPHandler | Obiekt do wysyłania odpowiedzi. |
| address | InetAddress | Adres klienta. |
| port | int | Port klienta. |
| packet | DatagramPacket | Otrzymany pakiet. |

* **Metody:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nazwa** | **Typ zwracany** | **Parametry** | **Opis** |
| run() | void | - | Sprawdza w osobnym wątku, czy otrzymany pakiet to zapytanie "CCS DISCOVER" i wysyła odpowiedź "CCS FOUND". |

Klasa TCPHandler

Klasa obsługująca operacje wysyłania wiadomości w protokole TCP.

* **Pola:**

|  |  |  |
| --- | --- | --- |
| **Nazwa** | **Typ** | **Opis** |
| socket | Socket | Połączenie TCP z klientem. |

* **Metody:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nazwa** | **Typ zwracany** | **Parametry** | **Opis** |
| getMessage() | String | - | Odbiera wiadomość z połączenia TCP. |
| sendMessage(String message) | void | String message – wiadomość do wysłania. | Wysyła wiadomość do klienta. |
| close() | void | - | Zamyka połączenie TCP. |
| isOpened() | boolean | - | Sprawdza, czy połączenie TCP jest otwarte. |

Klasa TCPServer

Serwer TCP, który nasłuchuje połączeń i deleguje zadania do obsługi.

* **Pola:**

|  |  |  |
| --- | --- | --- |
| **Nazwa** | **Typ** | **Opis** |
| port | int | Numer portu serwera. |
| statisticsHandler | TCPStatisticsHandler | Obiekt do zbierania statystyk. |

* **Metody:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nazwa** | **Typ zwracany** | **Parametry** | **Opis** |
| run() | void | - | Uruchamia serwer TCP w wątku, akceptując połączenia i przekazując je do obsługi. |

Klasa TCPStatisticsHandler

Zbieranie i zarządzanie statystykami serwera TCP.

* **Pola:**

|  |  |  |
| --- | --- | --- |
| **Nazwa** | **Typ** | **Opis** |
| allStatistics | ConcurrentHashMap<String, Integer> | Zbiór statystyk dla całego działania (zoptymalizowane pod kątem wielowątkowości). |
| lastStatistics | ConcurrentHashMap<String, Integer> | Statystyki dla ostatniego okresu (zoptymalizowane pod kątem wielowątkowości). |
| keys | String[] | Klucze dla statystyk. |
| allUniqueAddress | ConcurrentSkipListSet<String> | Przechowuje unikalne identyfikatory klientów (adres:port) (zoptymalizowane pod kątem wielowątkowości). |
| lastUniqueAddress | ConcurrentSkipListSet<String> | Przechowuje unikalne identyfikatory klientów (adres:port) z ostatniego okresu (zoptymalizowane pod kątem wielowątkowości). |

* **Metody:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nazwa** | **Typ zwracany** | **Parametry** | **Opis** |
| changeStatistics(String key, int value) | void | - | Zmienia statystyki na podstawie klucza. |
| changeClientStatistics(String address, String port) | void | - | Tworzy unikalny identyfikator klienta (adres:port) i dodaje je do zbioru unikalnych elementów. |
| resetLastStatistics() | void | - | Resetuje statystyki dla ostatniego okresu. |
| printStatistics() | void | - | Drukuje wszystkie statystyki. |
| run() | void | - | Co 10 sekund drukuje statystyki i resetuje je. |

Klasa TCPTaskHandler

Klasa obsługująca zadania, która przetwarza zapytania operacyjne TCP (ADD, SUB, MUL, DIV).

* **Pola:**

|  |  |  |
| --- | --- | --- |
| **Nazwa** | **Typ** | **Opis** |
| address | String | Adres klienta. |
| port | String | Port klienta. |
| handler | TCPHandler | Obiekt do wysyłania odpowiedzi do klienta. |
| statisticsHandler | TCPStatistics | Statystyki operacji. |
| pattern | String | Wzorzec dla operacji. |

* **Metody:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nazwa** | **Typ zwracany** | **Parametry** | **Opis** |
| run() | void | - | Przetwarza odebrane operacje matematyczne i wysyła wynik do klienta. |

Logika

Serwer UDP

Serwer nasłuchuje na porcie określonym przy uruchomieniu aplikacji. Obsługiwane są pakiety UDP. Klient wysyła wiadomość do serwera w formacie tekstowym przekonwertowanym na tablicę bajtów. Wiadomość ta musi być dokładnie zapisana jako ciąg znaków „CCS DISCOVER”. Maksymalna długość odbieranego pakietu to 12 bajtów. Jeśli klient wyśle wiadomość, której długość przekracza 12 bajtów, serwer odczyta tylko pierwsze 12 bajtów, a reszta danych zostanie zignorowana. Na poprawnie sformułowaną wiadomość serwer odpowiada wiadomością tekstową “CCS FOUND” na adres i port, z którego przyszło zapytanie. Każdy odebrany pakiet jest obsługiwany przez osobny wątek.

Serwer TCP

Dla każdego nowego klienta tworzony jest oddzielny wątek obsługujący komunikację. Wiadomości od klienta są przesyłane w formacie tekstowym „ADD 5 3\n” i odczytywane znak po znaku, do momentu napotkania znaku nowej linii „\n”, co oznacza koniec wiadomości. Serwer sprawdza poprawność wiadomości ze wzorcem przy użyciu wyrażenia regularnego: „^(ADD|SUB|MUL|DIV) -?\\d+ -?\\d+$”. Na podstawie nazwy operacji serwer wykonuje odpowiednie działanie matematyczne. Wynik operacji (lub komunikat „ERROR” w przypadku błędu) jest wysyłany do klienta w formie tekstowej zakończonej znakiem nowej linii „\n”. W przypadku zakończenia komunikacji przez klienta, serwer zamyka gniazdo.

Serwer prowadzi statystyki, obsługiwane przez osobny wątek. Są one aktualizowane w czasie rzeczywistym oraz wyświetlane co 10 sekund.

Ograniczenia

Każdy klient uruchamia nowy wątek w TCPServer, co przy dużej liczbie klientów może wyczerpać zasoby systemowe.