Dokumentacja Zad 1 z przedmiotu SKJ (s30593)

Co zostało zrealizowane?

- Program poprawnie uruchamia się w trybie master bądź slave, w zależności od tego czy port przekazany jako parametr jest już zajęty.
- Program w przypadku przekazania błędnych parametrów zgłasza błąd oraz kończy pracę.
- Program w trybie master poprawnie wykonuje swoje operacje w zależności od otrzymanego numeru.
- Program w trybie slave poprawnie wysyła numer, oraz upewnia się, że master go otrzymał, w przeciwnym przypadku ponawia próbę 5 razy.

Opis zaimplementowanego protokołu:

Protokół może przesyłać jedynie dwa typy informacji:

- Potwierdzenie – jest to String o wartości „OK” który zostaje wysłany przez mastera po otrzymaniu liczby.
- Liczba – jest to String zawierający liczbę, która jest wysyłana przez slave-a do mastera która definiuje następne kroki masterowi, bądź obliczona średnia przez mastera broadcastowana do lokalnej sieci lub informacja o zakończeniu działania programu.

Opis klas w programie:

- DAS – główna klasa programu, zawiera „entry point”.
- Master – klasa zarządzająca trybem master tego programu
- Slave – klasa zarządzająca trybem slave tego programu
- UDPManager – klasa pomocnicza do wysyłania i odbierania pakietów

Opis działania programu:

- main() – Metoda sprawdza poprawność przekazany argumentów, w przypadku błędnych, bądź niedostatecznych informuje o tym użytkownika i kończy pracę programu. Następnie w zależności czy port przekazany jako argument jest poprawny uruchamia program w odpowiednim trybie:
  - Master.work() – w przypadku sprawdzanego wcześniej portu, jeżeli jest on wolny to uruchamia program w trybie master
  - Slave.work() – natomiast jeżeli sprawdzany wcześniej port jest zajęty, przechodzi on w tryb slave
- Master.work(UDPManager, manager, startNumber) – główna metoda zarządzająca działaniem aplikacji w trybie mastera. Jej funkcjonalność opiera się na następujących krokach:
  - setBroadcastingAddress() – Używamy tej metody do zainicjalizowania zmiennej broadcastAddress która przechowuje adres rozgłoszeniowy aktualnej sieci. Metoda ta iteruje po wszystkich interfejsach sieciowych dostępnych w naszym systemie i wybiera pierwszy który nie jest loopback-iem, wirtualnym czy nie działającym interfejsem. I pobiera jego adres rozgłoszeniowy.

    - recievedNums – zmienna, która będzie przechowywać otrzymane liczby różne od 0 i -1 oraz pierwszą liczbę przekazaną jako arguement
    - Następnie w pętli zaczynamy nasłuchiwać pakietów przychodzących, jeżeli otrzymany pakiet ma treść „OK” pomijamy go, gdyż master nie potrzebuje żadnych potwierdzeń. Sprawdzamy wartość zmiennej sentBroadcast która jest prawdziwa, gdy został wysłany broadcast do lokalnej sieci. Jeżeli ta wartość jest prawdziwa to odsyłamy potwierdzenie „OK”. W zależności, gdy to jest liczba następne rzeczy się dzieją:
        - 0 – obliczana jest średnia ze wszystkich liczb zapisanych w zmiennej recievedNums, za pomocą stream-u jeżeli lista jest pusta to ustawiamy średnią jako 0. Następnie ta średnia jest wypisywana oraz broadcastowana do sieci lokalnej za pomocą metody broadcastToLan().
        - -1 – jest to komunikat o zakończeniu programu, liczba to jest wypisywana na ekran, broadcastowana do sieci lokalnej za pomocą metody broadcastToLan() a następnie program się kończy.
        - Inna liczba – sprawdzamy czy zmienna sentBroadcast jest prawdziwa, jeżeli nie jest to zapisujemy tą liczbę w tablicy recvNums oraz ją wypisujemy, następnie ustawiamy wartość zmienną sentBroadcast na fałsz.
- Slave.work() - główna metoda zarządzająca działaniem aplikacji w trybie slave. Jej funkcjonalność opiera się na następujących krokach:

    W pętli zależnej od zmiennej foundPort gdy jest fałszywa, próbujemy otworzyć socket na porcie o wartości MIN_PORT w zależności czy nam się uda czy nie następne rzeczy się dzieją:
        - Jeżeli się nie uda, losujemy nowy port w zakresie od MIN_PORT do MAX_PORT i wracamy do kroku próby otwarcia socketu.
        - Jeżeli nam się uda otworzyć port, ustawiamy wartość foundPort na fałsz oraz w pętli wysyłamy numer przekazany jako argument. Następnie oczekujemy na odpowiedź mastera, jeżeli jej nie otrzymamy po 2 sek wyświetlamy o tym informacje oraz ponawiamy wysłanie (maksymalnie 5 razy), w przeciwnym razie informujemy, że informacja nie została wysłana, wychodzimy z pętli i kończymy program.
- UDPManager.send(String msg) – pomocnicza metoda do wysyłania pakietów o treści msg. Upewnia się czy wiadomość nie jest za długa (max 10 bajtów), tworzy pakiet i wysyła go.
- UDPManager.recive() – pomocnicza metoda do otrzymywania pakietów, tworzy bufor o rozmiarze 10 bajtów, odbiera pakiet, następnie zwraca String-a z zawartością pakietu.