DOKUMENTACJA PROGRAMU

Aby uruchomić program należy mieć zainstalowaną Jave w wersji 1.8 lub nowszej oraz IDE IntelliJ.

W celu uruchomienia programu:

1. Należy wypakować pobrany zip z projektem
2. Uruchomić IntelliJ i otworzyć projekt „KnockKnock" znajdujący się w wypakowanym folderze „SKJprojekt". Aby to zrobić: File -> Open -> A następnie wybrać ścieżkę do zapisanego projektu i kliknąć „OK"
3. Następnie wejść w Run/Debug Configuration, kliknąć „+" -> "Application", wybrać "Main class" z klasy Server. W "Program arguments" wpisać oddzielone spacją porty, na których serwer ma nasłuchiwać pakietów i kliknąć „OK".
4. Uruchomić Server klikając  .
   Na konsoli, w ostatniej linii ukaże się nazwa hosta oraz adres IP. Należy skopiować sam adres IP.
5. Następnie ponownie wejść w Run/Debug Configurations, kliknąć „+" -> "Application", wybrać "Main class" z klasy Client. W "Program arguments" wkleić adres IP serwera oraz po spacji, również oddzielone spacją, porty, na które klient ma wysyłać pakiety.
6. Uruchomić Client klikając  .

Program został uruchomiony.

Opis użytych klas

I. Klasa Server

Otwiera porty UDP podane w argumentach programu, na których będzie nasłuchiwać na pakiety od klientów. Jeśli zostanie wykryta odpowiednia sekwencja pakietów UDP wysyłanych z tego jednego adresu, otwiera losowo wybrany port TCP i na adres z którego te pakiety przychodziły, wysyła komunikat UDP z numerem tego portu TCP. Następnie oczekuje połączenia TCP i po zakończeniu prostej komunikacji rozłącza się, po czym ponownie zaczyna nasłuchiwać na portach UDP.

- void setKeySequence(List<Integer> keySequence) – inicjalizuje listę keySequence, stanowiącą w jakiej kolejności i na jakie porty klient ma „zapukać"
- List<Integer> getKeySequence() – zwraca listę, na jakie porty i w jakiej kolejności mają zostać wysłane pakiety
- Map<String,List<Integer>> getClientsKnocking() – zwraca mapę zawierającą adresy klientów oraz porty do których wysłali pakiety
- void addNewClientKnocking(String key, int data) – dodaje do mapy klientów nowego klienta
- void addNewPacketFromClient(String key, int data) – dodaje do klienta o danym kluczu key (adresIP:port) data, czyli port, na który wysłał pakiet
- boolean clientKnocked(String key) – zwraca true jeśli w mapie klientów znajduje się klient o podanym key (adresIP:port)
- void sendTCPport(DatagramSocket socket, InetAddress clientAddress, int clientPort) – otwiera losowy port TCP i wysysła w odpowiedzi do klienta numer portu TCP
- void service(DatagramSocket socket) – odbiera pakiety I uruchamia ServerThread
- void listenSocket(int port) – zaczyna nasłuchiwanie na podanym porcie, wywołuje w pętli, do momentu powstania wyjątku, metodę service()

- main – sprawdza liczbę argumentów, tworzy obiekt Server, ustanawia oczekiwaną sekwencję pakietów, która umożliwi połączenie TCP, w wątkach wywołuje metodę listenSocket() dla każdego podanego portu.

A. Klasa ServerThread

Wewnętrzna klasa Server, dziedziczy po Thread.

- run() – wypisuje treści przychodzących pakietów na dane gniazdo serwera UDP, sprawdza czy adres klienta jest już zapisany w mapie klientów, jeśli nie to dodaje go i port do którego wysłał pakiet, a jeśli tak, to uaktualnia listę portów na które ten klient wysłał pakiety dopisując kolejny. Jeśli klient wysłał pakiety w oczekiwanej sekwencji to zostaje uruchomiona metoda sendTCPport()

II. Klasa TCPServer

Otwiera port TCP, na którym oczekuje na nawiązanie połączenia przez klienta.

- void listenSocket(int port) – zaczyna nasłuchiwanie na danym porcie

A. Klasa ServerThread

Wewnętrzna klasa TCPServer, dziedziczy po Thread

- run() – wysyła pakiet, odbiera i kończy połączenie

III. Klasa Client

Aplikacja przyjmuje jako parametry adres serwera i numery portów, na które wysyła pakiety. Po uruchomieniu otwiera port UDP, z którego wysyła serię pakietów po kolei na wszystkie porty. Po ich wysłaniu oczekuje na odpowiedź zawierającą numer portu TCP, z którym to następnie nawiązuje połączenie. Po wykonaniu prostej komunikacji z serwerem TCP rozłącza się i kończy pracę. Jeśli sekwencja pakietów UDP jest niepoprawna, kończy pracę z komunikatem „Error during communication”.

- void sendMsg(String msg) – wysyła pakiet UDP o treści msg
- String getResponse() – odbiera pakiet i zwraca odpowiedź od Servera w postaci Stringa
- void close() – zamyka gniazdo klienta
- void connectToTCP(int port) – tworzy obiekt klasy TCPClient I nawiązuje połączenie z serwerem TCP na podanym porcie port.
- main() – inicjalizuje adres IP ip pierwszym argumentem wywołania programu. Do każdego portu, podanego w następnych argumentach wywołania, wysyła pakiet UDP. Następnie czeka na odpowiedź od serwera i jeśli ją dostanie to nawiązuje połączenie z serwerem TCP i po uzyskaniu od niego odpowiedzi kończy działanie.

IV. Klasa TCPClient

Klasa służąca do nawiązywania połączenia z Serwerem TCP.

- void connectToTCPServer(int port, String address) – tworzy gniazdo i próbuje połączyć się z serwerem TCP na zadanym adresie IP address i porcie port. Wysyła zapytanie i odczytuje odpowiedź, po czym kończy działanie.