

### Wieloużytkownikowy serwer ECHO (TCP)

#### Cel ćwiczenia:

W trakcie ćwiczenia studenci wykonają pierwszy prosty program sieciowy wieloużytkownikowego serwera usługi ECHO.

Usługa Echo: serwer czeka na połączenie klienta. Po zaakceptowaniu połączenia oczekuje na wiadomość. Zaimplementowany program serwera można testować napisaną na poprzednich zajęciach własną aplikacją klienta bądź aplikacją dostępną w folderze *PROGRAMY* w sekcji „Połączenie TCP klient-serwer”.

#### Polecenie ćwiczeniowe:

Do zaliczenia ćwiczenia wymagane jest napisanie programu udostępniającego kilku użytkownikom jednocześnie usługę ECHO. Możliwa dowolna platforma programistyczna.

Zadaniem programu jest:

- udostępnianie usługi dla wybranych interfejsów sieciowych (**domyślnie dla wszystkich, a nie localhost!**),
- udostępnienie usługi ECHO dla protokołu TCP/IP na podanym porcie (wartość domyślna 7 ).
- udostępnienie możliwości wprowadzenia innego portu komunikacji,
- rejestrowanie wszystkich połączeń i wyświetlanie dostępnych informacji o podłączonym komputerze odległym wraz z numerem portu tej komunikacji,
- posiadać limit aktywnych połączeń do 3, a po jego przekroczeniu wysyłać do kolejnego nadliczbowego klienta komunikat o zajętości serwera i rozłączać połączenie,
- odporność na niewłaściwe dane wpisane przez użytkownika, niewłaściwe zamknięcie gniazda po stronie klienta i zwalnianie zasobów (wątków) w chwili zamykania programu,
- natychmiastowe odsyłanie wiernej kopii otrzymanej wiadomości (tzn. funkcja wysyłająca wysyła TYLKO tyle danych, ile serwer otrzymał od klienta),
- informowanie użytkownika o każdej otrzymanej wiadomości z zaznaczeniem IP i numeru portu nadawcy oraz treści i rozmiaru,
- informowanie użytkownika o bieżącej liczbie połączonych klientów.

Algorytm programu powinien być zgodny z tym przedstawionym na diagramie na następnej stronie:

#### UWAGA!!

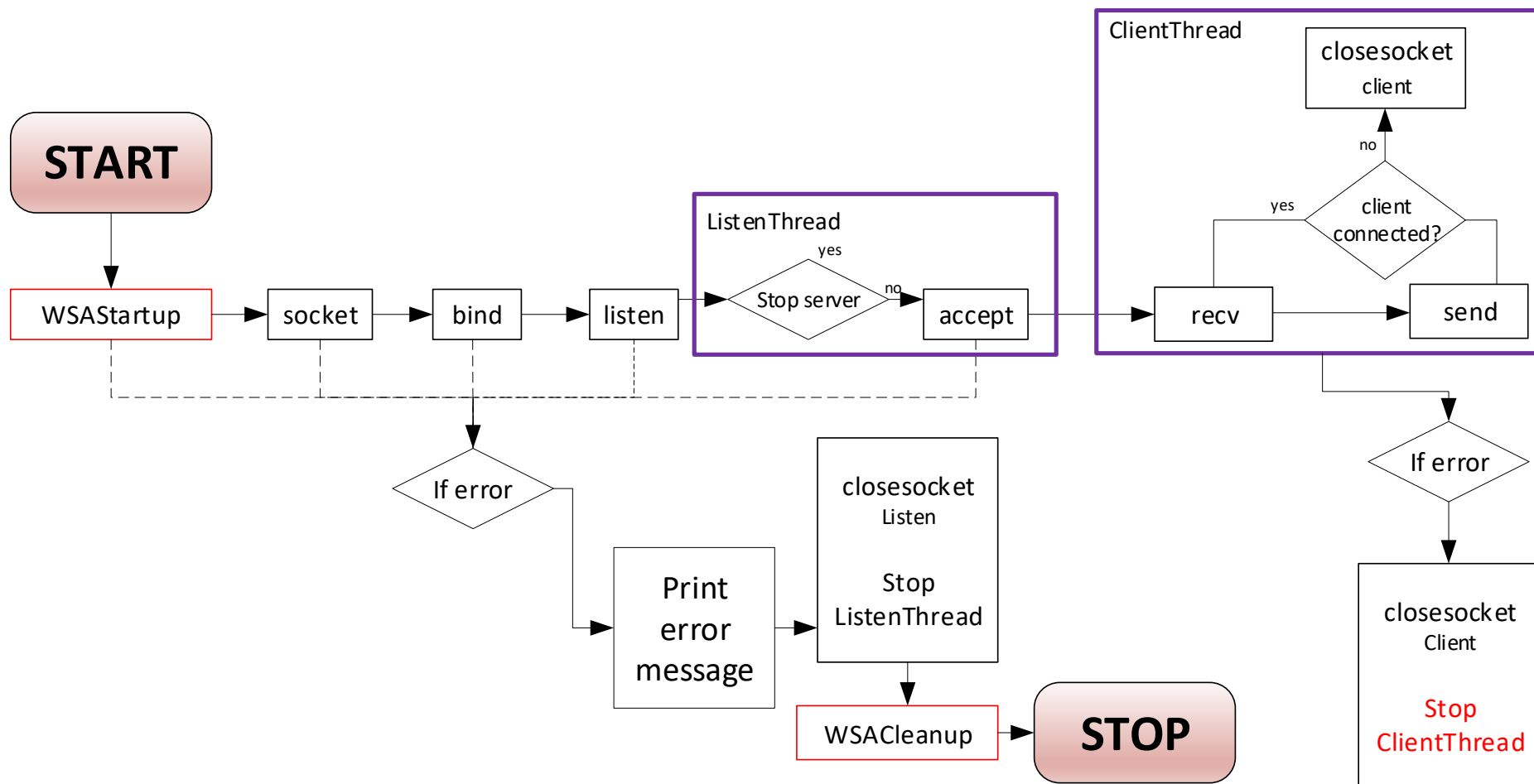
Program można zrealizować na dwa sposoby. Pierwszy, to implementacja poszczególnych bloków programu (np. oczekiwanie na połączenie klienta lub oczekiwanie na wiadomość od klienta) w oparciu o wątki. Inną możliwością jest wykorzystanie metod asynchronicznych do obsługi blokujących operacji komunikacji sieciowej.

#### Uwagi do nadsyłanych programów

1. Program powinien spełniać wszystkie kryteria opisane powyżej i być wynikiem samodzielnej pracy,
2. Dozwolone jest użycie różnych języków programowania przy założeniu, że każda z operacji zaznaczonych na diagramie zostanie zaimplementowana (wywołana / obsłużona) z osobna.  
Wyjątek w tym przypadku stanowią bloki WSStartup i WSACleanup.  
Wykonanie zadania w oparciu o automatyzację wynikającą z wywołania jedynie metody konstruktora spowoduje obniżenie oceny o 1.
3. Zaimplementowana procedura obsługi błędów i sytuacji wyjątkowych (jej brak obniża ocenę o jeden punkt),
4. Interfejs użytkownika dowolny tj. konsola, GUI.
5. Czytelny i przejrzysty kod uwzględniający wysoki poziom hermetyzacji i synchronizację wątków.

#### Odwołania

1. C#: <https://social.msdn.microsoft.com/Forums/en-US/d9e4bcb3-3149-4817-8d7c-37c37c37c37c/tcp-server-with-multiple-clients?forum=nc1>
2. JAVA: <http://tutorials.jenkov.com/java-multithreaded-servers/multithreaded-server.html>



Bloki zaznaczone na czerwono są wymagane tylko w przypadku implementacji w języku C/C++.