Programowanie sieciowe

Laboratorium 2 **dr hab. inż. Radosław Wajman**

Iteracyjny serwer ECHO (TCP)

Cel ćwiczenia:
W trakcie ćwiczenia studenci wykonają program iteracyjnego serwera usługi ECHO na bazie protokołu TCP.

Usługa Echo: serwer czeka na połączenie klienta. Po zaakceptowaniu połączenia oczekuje na wiadomość. Każdą odebraną od klienta wiadomość natychmiast odsyła do klienta bez zmian.
Zaimplementowany program serwera można testować napisaną na poprzednich zajęciach własną aplikacją klienta *TCP_Client.jar* bądź aplikacją dostępną w folderze PROGRAMY.

Polecenie ćwiczeniowe:
Do zaliczenia ćwiczenia wymagane jest napisanie programu udostępniającego usługę ECHO. Do realizacji ćwiczenia można zastosować dowolną preferowaną platformę programistyczną.

Zadaniem programu jest:

- nasłuchiwanie usługi dla wybranych interfejsów sieciowych (domyślnie dla wszystkich, a nie tylko localhost!),
- udostępnienie usługi ECHO dla protokołu TCP/IP na podanym porcie (wartość domyślna 7).
- udostępnienie możliwości wprowadzenia innego portu komunikacji,
- rejestrowanie wszystkich połączeń klientów, przypisywanie im numeru # i wyświetlanie w stałym miejscu (znajdującym się 80% od lewej krawędzi okna oraz 10% od górnej krawędzi) dostępnych informacji o podłączonym komputerze odległym – adres IP wraz z numerem portu tej komunikacji, np. #1 192.168.0.10:34567
- odporność na niewłaściwe dane wpisane przez użytkownika, niewłaściwe zamknięcie gniazda po stronie klienta i zwalnianie zasobów w chwili zamykania programu w tym zasobów związanych z aktualnie podłączonym klientem,
- 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 numeru # przypisanego nadawcy wiadomości oraz treści i rozmiaru wiadomości.
- informowanie o akceptacji nowego połączenia oraz rozłączenia klienta,

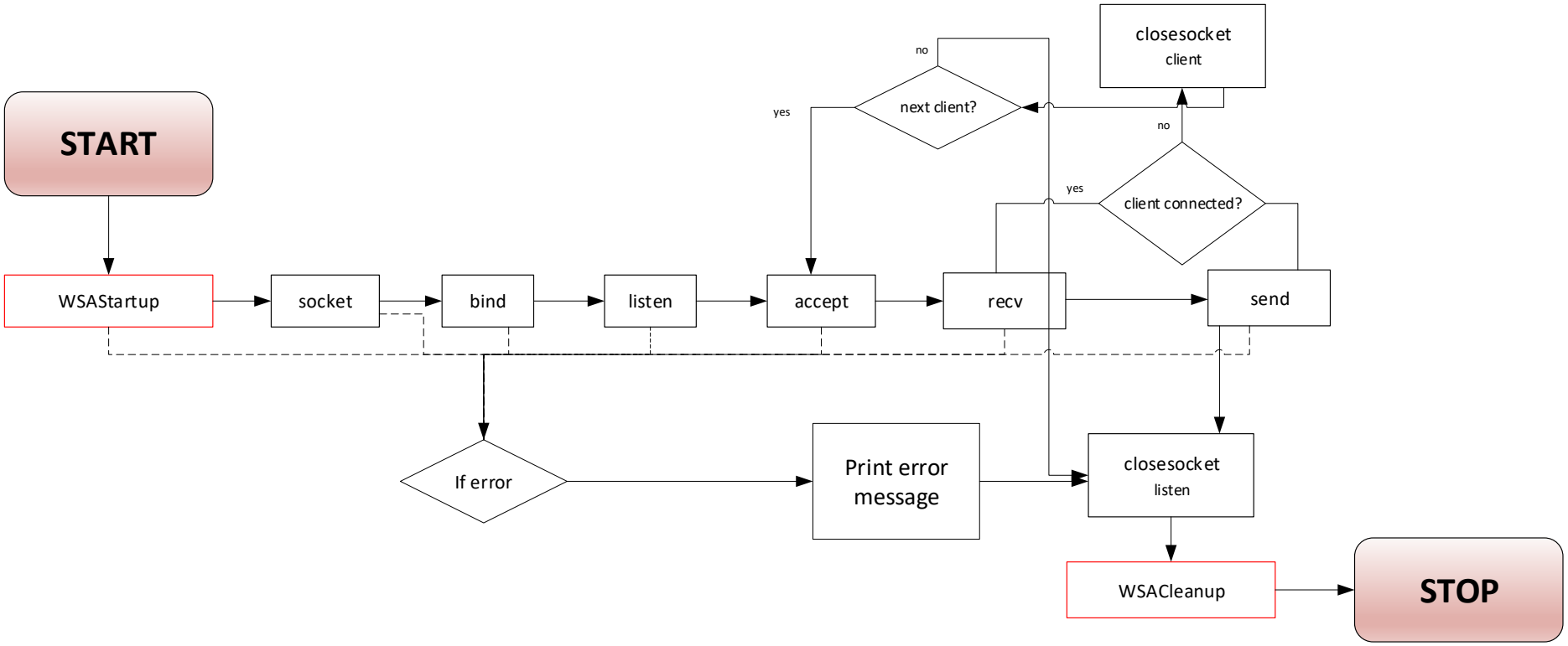
Algorytm programu powinien być zgodny z tym przedstawionym na diagramie na następnej stornie:

Uwagi do nadsyłanych programów

1. Program powinien spełniać wszystkie funkcjonalności opisane powyżej i żadnej więcej.
2. Powinien być wynikiem samodzielnej pracy,
3. 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 WSAStartup i WSACleanup.
   Wykonanie zadania w oparciu o automatyzację wynikającą z wywołania jedynie metody konstruktora spowoduje obniżenie oceny o 1.
4. Zaimplementowana procedura obsługi błędów i sytuacji wyjątkowych (jej brak obniża ocenę o jeden punkt),
5. Interfejs użytkownika dowolny tj. konsola, GUI.
6. Czytelny i przejrzysty kod uwzględniający wysoki poziom hermetyzacji.

Odwołania

1. C: http://www.binarytides.com/winsock-socket-programming-tutorial/
2. C#: https://msdn.microsoft.com/en-us/library/system.net.sockets.socket.listen(v=vs.110).aspx
3. JAVA: https://systembash.com/a-simple-java-tcp-server-and-tcp-client/
4. JS: https://github.com/socketio/socket.io-client/blob/master/docs/API.md
5. Pyton: https://docs.python.org/3/howto/sockets.html



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