Serializacja

Wiele z serwerów pracujących np. w modelu REST jest w stanie udostępniać dane w wielu formatach, na przykład znanych już nam formatach JSON, oraz XML

W zadaniu tym spróbujemy napisać mechanizm obsługi serializacji wiadomości przez serwer.

Projekt zawiera:

1. Program.cs zawierający funkcję main
2. Interfejsy konkretnych wiadomości.

Założenia:

1. W celu uproszczenia wiadomości założymy, że wszystkie będą w następującym formacie: {NAZWA_FORMATU}treść wiadomości{NAZWA_FORMATU}. Elementy wewnątrz wiadomości będą podzielone konkretnym znakiem, np.: “/”. Proszę zwrócić uwagę, że to tylko uproszczony model. W rzeczywistości różne formaty będą miały kompletnie różne sposoby serializacji.
2. Metody odpowiedzialne za serializację powinny dodawać początkowy i końcowy tag.
3. Dodanie nowego rodzaju serializacji nie powinno powodować żadnych zmian w programie użytkownika.

Twoje zadanie:

1. Implementacja interfejsów wiadomości, opis poniżej.
2. Implementacja programu użytkownika, który nieświadom wybranego sposobu serializacji prawidłowo stworzy listę wiadomości. Szczegóły poniżej.
3. Implementacja funkcji main, która dla każdego z formatów uruchomi program użytkownika, a następnie wypisze wszystkie zwrócone wiadomości (ich metoda Serialize) na ekran.

Program użytkownika.

Napisz funkcję która udaje kod serwera stworzonego przy użyciu naszego frameworku. Ma ona stworzyć listę wiadomości nie znając wybranego sposobu serializacji.

Powinna ona:

- stworzyć wiadomości każdego typu i uzupełnić je (np OwnerName - John, OwnerStatus - Owner, etc).
- zwrócić listę zawierającą te wiadomości.

Opis formatów:

StandardFormat:

tagi: {SF}
elementy wiadomości oddzielone są znakiem “/”

ExtendedFormat:
tagi: {EF}
elementy wiadomości oddzielone są znakiem “$”

RandomFormat:
tagi: {RFM}
elementy wiadomości oddzielone są znakiem “&”

Wiadomości:

Wiadomości będą wyglądały różnie w zależności od wybranego formatu:

StandardFormat
- Każde z pól będzie normalnie zapisane (metodą ToString).

ExtendedFormat
- Każdy z napisów jest zamieniony na wielkie litery. wartości liczbowe otoczone są dodatkowymi znakami “?”, np: {EF}?125?{EF}

RandomFormat
- znaki w elementach wiadomości są odwrócone. Dodatkowo wszystkie liczby powinny być zanegowane.

Każdy interfejs ma metodę Serialize. Powinna ona zwrócić zawartość wiadomości w prawidłowym formacie (tagi, znak oddzielający). Każda z wiadomości zawiera inne pola.

Np wiadomość IFundsReport zawiera dwa pola liczbowe. Jej serializacja w StandardFormat będzie wyglądała tak:

{SF}125/4738.35{SF}

ale formatem RandomFormat już tak:

{RFM}521-&53.8374-{RFM}

IErrorMessage ma stałą zawartość: “Error encountered”, ale oczywiście w formacie RandomFormat powinna ona wyglądać tak: “deretnuocne rorrE”.