Informatyka w Mechatronice
Serwer danych pomiarowych

March 20, 2018

1 Cel ćwiczenia

Celem ćwiczenia jest implementacja w języku Java klas klienta i serwera danych pomiarowych przesyłanych w systemie monitorowania. W pierwszym kroku należy zbudować aplikację klienta umożliwiającego generowanie danych, przesyłanie ich do serwera i pobieranie z serwera zapisanych danych.

2 Wymagania dla wersji pierwszej

Należy zaimplementować aplikację klienta, działającą z wykorzystaniem protokołu UDP, który zapewni generowanie danych typu *TimeHistory*, *Spectrum* i *Alarm*, a następnie prześle te dane do serwera celem ich zapisu i udostępniania.

2.1 Wymagania dotyczące klienta

Aby zrealizować aplikację klienta można posłużyć się przykładem kodu klasy *UdpClient*, zawartej w pakiecie *networking.Udp* projektu *Lecture2*. Potrzebne będą następujące zmiany:

1. utworzenie nowego projektu typu 'Java' o dowolnej nazwie.
2. zaimportowanie z projektu *Lecture2* kod klas *UdpServer* i *UdpClient* zawartych w pakiecie *networking.Udp*
3. w celu wykorzystania klas *TimeHistory* i *Spectrum* w nowym projekcie należy włączyć projekt realizowany na poprzednim ćwiczeniu do nowego projektu. W tym celu należy w kontekście nowego projektu wykonać polecenie *Properties –> Build path –> Projects –> Add* i w okienku dialogowym wybrać projekt realizowany na poprzednim ćwiczeniu. Należy pamiętać o nadaniu pakietom w źródłowym projekcie nazw innych niż 'default', co pozwoli zaimportować klasy w nich zawarte
4. skopiować do nowo otworzonej klasy *Tools* kod metod *serialize(Object obj)* i *deserialize(byte[] bytes)* klasy *FileOperations* zawartej w pakiecie *main* projektu *Lecture2*
5. przejść do projektu realizowanego na poprzednim ćwiczeniu i zmodyfikować kod klasy *Packet*, dodając bezpośrednio za nazwą klasy frazę 'implements Serializable', co pozwoli dokonać serializacji i deserializacji obiektów jej konkretnych podklas
6. rozbudować metodę *main()* klasy *UdpClient* o następujące operacje:
   - utworzenie obiektu *packet* klasy *TimeHistory* lub *Spectrum*
   - serializacja obiektu do **tablicy bajtów** za pomocą instrukcji *byte[] data = Tools.serialize(packet);*
   - utworzenie obiektu klasy *DatagramPacket* zawierającego dane z tablicy *data*. UWAGA: należy się posłużyć konstruktorem z czterema parametrami, ponieważ oprócz tablicy bajtów i jej długości w pakiecie musi być zawarty docelowy adres IP i port
   - wysłanie pakietu do serwera za pomocą metody *send*
   - utworzenie obiektu *reply* klasy *DatagramPacket* przeznaczonego na dane jakie będą zwrócone przez serwer. UWAGA: należy się posłużyć konstruktorem z dwoma parametrami (referencja do tablicy bajtów i ilość jej elementów)
   - wywołanie metody *receive* w celu odbioru danych zwróconych przez serwer
   - zadeklarowanie zmiennej *read* typu *Packet* i deserializacja danych znajdujących się w tablicy bajtów obiektu *reply*.
   - wyświetlenie tekstowej reprezentacji zmiennej *read*

2.2 Wymagania dotyczące serwera

Aby zrealizować serwer można posłużyć się przykładem kodu klasy *UdpServer*, zawartej w pakiecie *networking.Udp* projektu *Lecture2*. Pierwsza wersja może być identyczna jak dyskutowana na wykładzie, jako że umożliwia ona odesłanie do klienta kopii nadesłanych danych.

2.3 Testowanie klas

Aby wykonać testowanie klas należy:

1. uruchomić serwer
2. uruchomić aplikację klienta

3 Wymagania dla wersji drugiej

Należy stworzyć aplikację serwera danych pomiarowych, odpowiednio modyfikując serwer wykorzystywany w wersji pierwszej. Wymagane będą funkcje:

1. zapis ramki danych typu *TimeHistory* i *Spectrum* w pliku o nazwie opartej na nazwie urządzenia i opisie kanału zawartych w klasie *Packet*
2. wyszukanie i przesłanie do klienta danych z wybranego urządzenia i kanału
3. zwrócenie listy danych spełniających kryterium wyszukiwania oparte na nazwie urządzenia, opisie kanału i czasie rejestracji
4. zastosowanie osobnych wątków dla realizacji w.w. zadań i objęcie sekcją krytyczną wykluczających się operacji

UWAGA: Wymagana jest realizacja punktu numer 1 dla uzyskania minimalnej oceny pozytywnej. Realizacja punktu numer 2 albo 3 umożliwi uzyskanie oceny 3.5, a realizacja obu - oceny 4.0. Realizacja wszystkich podnosi ocenę na 5.0.

3.1 Zmiany w kodzie klienta

Należy przenieść z klienta do serwera, z koniecznymi zmianami, kod realizujący ostatnie cztery zadania wymienione w punkcie 6 rozdziału 2.1.
Kod ten powinien realizować po stronie serwera zapis danych, zgodnie z p. 1 wymagań dla wersji drugiej. Tak więc należy wykonać:

- utworzenie obiektu *reply* klasy *DatagramPacket* przeznaczonego na dane jakie będą nadesłane do serwera. UWAGA: należy się posłużyć konstruktorem z dwoma parametrami.
- wywołanie metody *receive* w celu odbioru nadesłanych danych
- zapis nadesłanych danych do pliku o nazwie *name* opartej na nazwie urządzenia i opisie kanału, za pomocą instrukcji *Files.write(new File(name).toPath(), reply.getData())*
- zadeklarowanie zmiennej *read* typu *Packet* i deserializacja danych znajdujących się w tablicy bajtów obiektu *reply* do obiektu reprezentowanego przez zmienną *read*
- wyświetlenie tekstowej reprezentacji zmiennej *read*

Aby zrealizować punkty 2 i 3 wymagań dla wersji drugiej, należy utworzyć dodatkową klasę, n.p. *Request*, która będzie reprezentować żądanie przekazane od klienta do serwera. Klasa ta powinna mieć pola *type* i *value*, określające typ żądania i dodatkową wartość precyzującą warunki związane z nim. Klasa *Request* powinna dziedziczyć z klasy *Packet* i implementować interfejs *Serializable*.
Aby zrealizować punkt 4 wymagań dla wersji drugiej należy utworzyć dodatkowe klasy dziedziczące z klasy *Thread*.