Assignment - SW Engineering

Your Task

Your task is to create a middleware application that in real-time transforms the information about objects from the server and relays them at fixed time intervals to the client. The included SAAB.jar file is a server application that provides objects using a TCP/IP connection. A client is expected to connect via TCP/IP to the middleware application. Once connected, the middleware relays only the latest status of each unique object at predetermined intervals, ensuring the report reflects only the most current information.

The objects from the server have different types (1-3), where types 1 and 2 are of the same category, and type 3 belongs to a second category. In the output channel (to the client), the objects should be categorized according to type, category, and current distance from a certain point in a coordinate system. The objects' coordinates are updated over time; thus, the application shall relay the latest status to the output at fixed time intervals.

Requirements:

- Implement using C++.
- The incoming objects' categorizations shall be updated to reflect their type, category, and proximity to the Cartesian coordinates (150, 150).
- The application shall log errors to `std::clog`.
- Every one and two-thirds seconds (approximately 1.67 seconds), the middleware should output the objects' latest processed values in binary format, adhering to the detailed specifications provided in the *Client Application* section below.

Example Information Flow:

1. **Server to Middleware:**
   - Continuous real-time reading and processing of data from the Server.
2. **Middleware to Client:**
   - Fixed-interval reporting to the Client of the latest processed information on each object.

Server Application

The software package sent to you contains a server application that delivers data in text format via TCP/IP. The server application is configured to communicate on port 5463.

Configuration and Start of the Server

The server application configuration file looks like this:

```
SERVERPORT=5463             # Default server port
MAP=map.gif                 # Path to map
```

To start the server from the command prompt:

```
java -classpath SAAB.jar com.saabtech.server.SAABServer
```

Server Data Specification

Data delivered from the server looks like:

```
ID=<S64>;X=<S32>;Y=<S32>;TYPE=<U8>
```

The table below describes the data fields in detail.

Field	Type	Example data	Description
ID	S64	2691882127991893	ID number of the object
X	S32	119	X-coordinate for the object
Y	S32	227	Y-coordinate for the object
TYPE	U8	2	Type-id for the object

For the *Type* column above, “S64” stands for “signed 64-bit integer,” “S32” for “signed 32-bit integer,” and “U8” for “unsigned 8-bit integer”.

Example 1:

```
ID=2691882127991893;X=250;Y=150;TYPE=3
```

Example 2 - Two objects with the same category:

```
ID=2691882127234543;X=199;Y=230;TYPE=1
ID=2691882127221587;X=229;Y=310;TYPE=2
```

Client application

The client application will receive data through its standard input, with the middleware outputting data in binary format. Endianness is not specified. Configuration for communication is not applicable as the client receives data directly from the middleware’s standard output.

Client Data Specification

Once the client connects to the middleware application, data delivered from the middleware application every one and two-thirds seconds (~1.7s) shall start like this:

Parameter Name	Units	Data Type
preamble	`0xFE00`	32-bit unsigned int
count		32-bit unsigned int

`count` is the number of objects to directly follow. Each object after the preamble has the following format:

Parameter Name	Data Type
Id	64-bit int
X	32-bit int
Y	32-bit int
Type	8-bit unsigned int
Color	3 bytes

Where the 3-byte color sequence representation is one of:

- Red: `0x5B 0x31 0x6D`
- Yellow: `0x5B 0x33 0x6D`
- Blue: `0x5B 0x34 0x6D`

Objects are colored according to:

- Category 2 objects are yellow unless they are closer than *100* units from the designated coordinate, in which case they turn red.
- Type 1 objects: Blue unless closer than *75* from the designated coordinate then yellow and if closer than *50* then red.
- Type 2 objects: Blue unless closer than *50* from the designated coordinate then yellow.

Code example

The code snippet below demonstrates how to establish a TCP connection to the server application and read data from it using the ASIO library in C++.

```cpp
#include <iostream>
#include <string>

#include <asio.hpp>

int main() {
  asio::ip::tcp::iostream input("localhost", "5463");
  for (std::string str; std::getline(input, str);)
    std::cout << str << '\n';
}
```

The example necessitates the ASIO library and a toolchain compatible with C++11 or later.