Yeungnam University

Real Time Location System

Design Document

[Revision history]

Revision date	Version #	Description	Author
	0.01	First Documentation	
	0.02	Client Danger Warning 수정	

= Contents =

1. Introduction 4

2. Class diagram 5

3. Sequence diagram 17

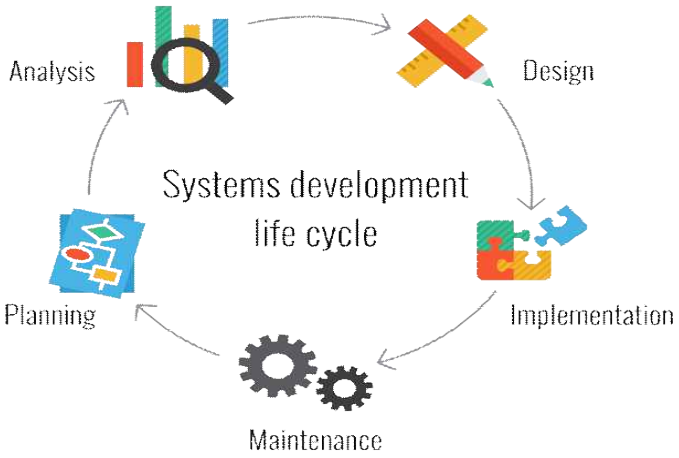
4. State machine diagram 30

5. Implementation requirements 32

6. Glossary 32

7. References 32

1. Introduction



<그림 1> Software Devlopment Life Cycle

본 문서는 Analysis Document의 다음 문서인 Design 단계의 문서로, 본 시스템에 대한 세 번째 문서이다. 이 단계에서는 Analysis 단계에서 분석한 Use case, Domain 등을 바탕으로 Class Diagram, Sequence Diagram, State Machine Diagram을 만들고 설명하는 것을 주로 다룬다.

Class Diagram

- "Class"라고 하는 객체지향 설계단위를 이용하여 시스템의 정적인 structure를 표현한 다이어그램
- Class의 Attributes, Methods 정의

Sequence Diagram

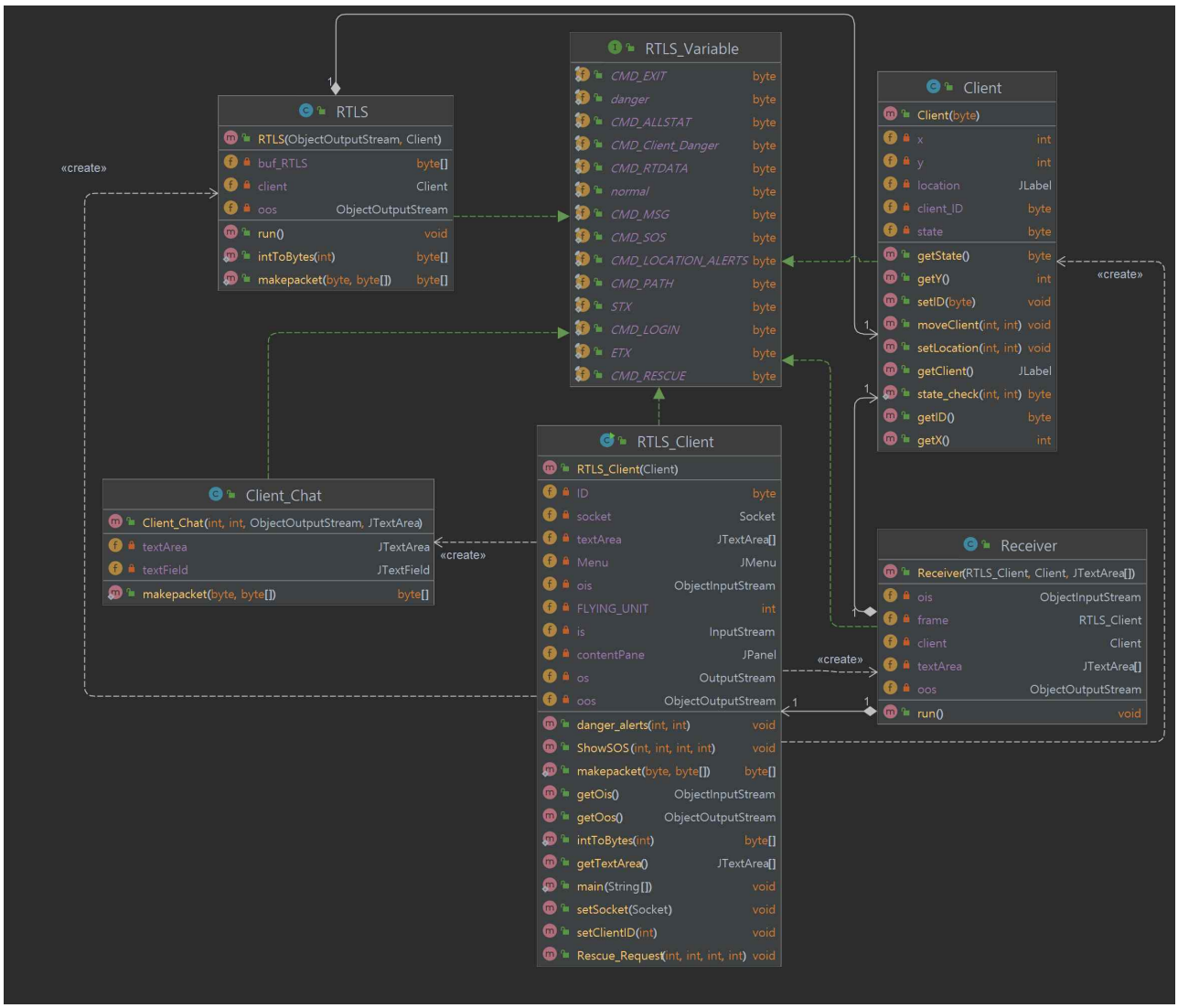
- 객체간의 동적 상호작용을 시간적 개념을 중심으로 모델링하는 과정
- 객체의 attribute, operation을 상세히 정의

State Machine Diagram

- 객체 lifetime동안의 변환될 수 있는 모든 상태를 정의해둔 다이어그램
- 상태와 그 상태가 전이하는 타이밍, 각 상태에 대한 event 기술

2. Class diagram

2.1. Client



<그림 2> Client Class Diagram

2.1.1. RTLS_Variable

Attributes
+STX:byte = 0x02 : 데이터의 시작을 의미하는 값 +ETX:byte = 0x03 : 데이터의 끝을 의미하는 값 +CMD_RTDATA:byte = 0x00 : 클라이언트의 위치와 상태 데이터를 의미하는 값 +CMD_ALLSTAT:byte = 0x01 : 모든 클라이언트의 상태 데이터를 의미하는 값 +CMD_MSG:byte = 0x02 : 메시지 데이터를 의미하는 값 +CMD_LOGIN:byte = 0x10 : 새로운 클라이언트 로그인 데이터를 의미하는 값 +CMD_Client_Danger:byte = 0x11 : 클라이언트 위험 알림 데이터를 의미하는 값 +CMD_SOS:byte = 0x33 : SOS 요청 데이터를 의미하는 값 +CMD_RESCUE:byte = 0x34 : 구조요청 데이터를 의미하는 값 +CMD_LOCATION_ALERTS:byte = 0x35 : 위치 경고 데이터를 의미하는 값 +CMD_EXIT:byte = 0x44 : 클라이언트 종료 데이터를 의미하는 값 +normal:byte = 0x00 : 클라이언트의 상태가 정상이라는 것을 의미하는 값 +danger:byte = 0xFF : 클라이언트의 상태가 위험하다는 것을 의미하는 값
Methods

2.1.2. Client

Attributes
-client_ID:byte : 클라이언트 ID -state:byte = 0x00 : 클라이언트 상태 -x:int : 클라이언트 x위치 -y:int : 클라이언트 y위치 -location:JLabel : 클라이언트 화면 패널에 현재 위치를 보여줄 라벨
Methods
+Client(ID:byte) : 생성자에서 ID를 설정 +getID() : 클라이언트 ID를 가져옴 +setID(ID:byte) : 클라이언트 ID를 설정 +getState() : 클라이언트 상태를 가져옴 +getX() : 클라이언트 x위치를 가져옴 +getY() : 클라이언트 y위치를 가져옴 +setLocation(x:int, y:int) : 클라이언트 x,y 위치를 설정 +getClient() : 클라이언트 라벨을 가져옴 +moveClient(moveX:int,moveY:int) : 클라이언트 라벨 위치를 조정

2.1.3. RTLS_Client

Attributes
−ID:byte : 클라이언트 ID를 저장하는 변수 −FLYING_UNIT:int : 클라이언트가 한번 움직이는 거리 −contentPane:JPanel : 클라이언트 화면 패널 −Menu:JMenu : 클라이언트 메뉴 공간 −socket:Socket : 서버와 연결된 소켓 −os:OutputStream : 서버에게 데이터 전송에 사용할 변수 −oos:ObjectOutputStream : 서버에게 데이터 전송에 사용할 변수 −is:InputStream : 서버에게 데이터를 받을 때 사용할 변수 −ois:ObjectInputStream : 서버에게 데이터를 받을 때 사용할 변수 −textArea:JTextArea [] : 클라이언트의 각 채팅 채널의 메시지를 보여줄 공간
Methods
+RTLS_Client(client:Client) : 생성자에서 클라이언트 객체 설정 +getOis() : ObjectInputStream 가져오기 +getOos() : ObjectOutputStream 가져오기 +getTextArea() : TextArea 가져오기 +setClientID(ID:int) : 클라이언트 화면 패널의 제목을 ID로 설정 +setSocket(socket:Socket) : 소켓 변수를 설정 +ShowSOS(ID:int, X:int, Y:int, State:int) : 클라이언트의 SOS요청 알림을 띄움 +danger_alerts(X:int, Y:int) : 옵저버의 위험지역 알림을 띄움 +Rescue_Request(ID:int, X:int, Y:int, State:int) : 옵저버의 구조요청 알림을 띄움

2.1.4. RTLS

Attributes
−buf_RTLS:byte[] : 현재 클라이언트의 위치 상태 정보를 담을 변수 −oos:ObjectOutputStream : 서버에게 데이터 전송에 사용할 변수 −client:Client : 클라이언트의 위치와 상태를 얻을 클라이언트 객체
Methods
+run() : 1초마다 현재 클라이언트의 위치, 상태를 서버로 전송한다. +RTLS(oos:ObjectOutputStream, client:Client) : 서버로 전송할 ObjectOutputStream과 Client 객체를 설정한다.

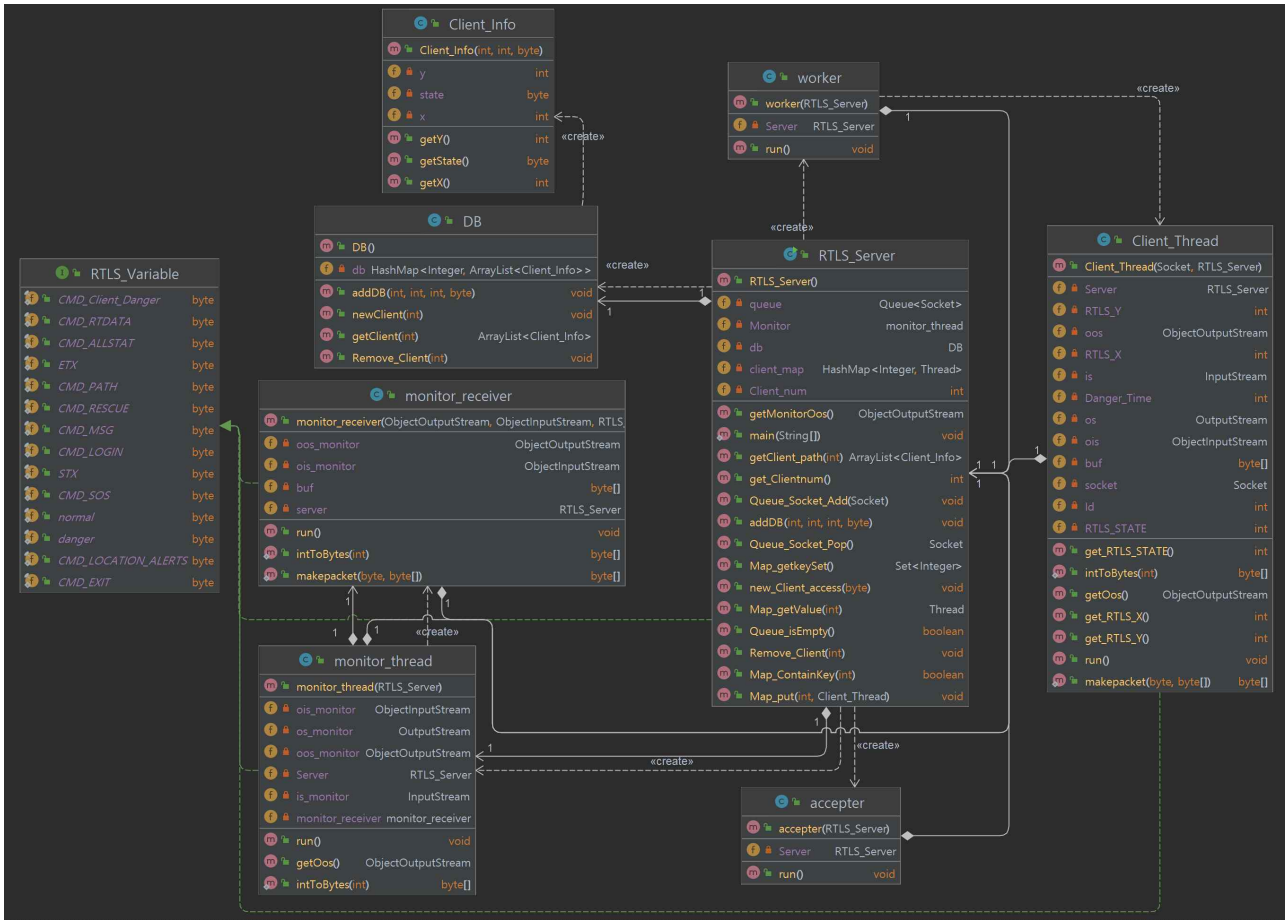
2.1.5. Receiver

Attributes
−client:Client : 클라이언트의 위치와 상태를 바꿀 클라이언트 객체 −textArea:JTextArea [] : 메시지를 적어 넣을 JTextArea −ois:ObjectInputStream : 서버에게 데이터를 받을 때 사용할 변수 −oos:ObjectOutputStream : 서버에게 데이터 전송에 사용할 변수 −frame:RTLS_Client : 클라이언트 화면 패널
Methods
+run() : 서버에서 데이터가 올 때마다 데이터를 처리함 +Receiver(frame:RTLS_Client, client:Client, textArea:JTextArea []) : 생성자에서 클라이언트 화면패널, 클라이언트 객체, JTextArea를 설정함

2.1.6. Client_Chat

Attributes
−textField:JTextField : 채팅 채널의 메시지를 보여줄 공간 −textArea:JTextArea : 채팅을 입력할 공간
Methods
+run() : 서버에서 데이터가 올 때마다 데이터를 처리함 +Client_Chat(to:int, ID:int, oos:ObjectOutputStream, TextArea:JTextArea) : 생성자에서 클라이언트 채팅 화면패널, oos변수 설정함

2.2. RTLS_Server



<그림 3> Server Class Diagram

2.2.1. RTLS_Variable

Attributes
+STX:byte = 0x02 : 데이터의 시작을 의미하는 값 +ETX:byte = 0x03 : 데이터의 끝을 의미하는 값 +CMD_RTDATA:byte = 0x00 : 클라이언트의 위치와 상태 데이터를 의미하는 값 +CMD_ALLSTAT:byte = 0x01 : 모든 클라이언트의 상태 데이터를 의미하는 값 +CMD_MSG:byte = 0x02 : 메시지 데이터를 의미하는 값 +CMD_LOGIN:byte = 0x10 : 새로운 클라이언트 로그인 데이터를 의미하는 값 +CMD_Client_Danger:byte = 0x11 : 클라이언트 위험 알림 데이터를 의미하는 값 +CMD_SOS:byte = 0x33 : SOS 요청 데이터를 의미하는 값 +CMD_RESCUE:byte = 0x34 : 구조요청 데이터를 의미하는 값 +CMD_LOCATION_ALERTS:byte = 0x35 : 위치 경고 데이터를 의미하는 값 +CMD_EXIT:byte = 0x44 : 클라이언트 종료 데이터를 의미하는 값 +normal:byte = 0x00 : 클라이언트의 상태가 정상이라는 것을 의미하는 값 +danger:byte = 0xFF : 클라이언트의 상태가 위험하다는 것을 의미하는 값
Methods

2.2.2. RTLS_Server

Attributes
−queue:Queue<Socket> : 새로운 클라이언트가 접속요청할 때 임시로 넣어주는 큐 −client_map:HashMap<Integer, Thread> : 클라이언트들을 관리하는 해시맵 −Client_num:int : 클라이언트의 수를 저장하는 변수 −Monitor:monitor_thread : 옵저버에게 1초마다 클라이언트 위치를 보내는 스레드 −db:DB : 데이터 베이스 클래스
Methods
+Queue_Socket_Add(socket:Socket) : 새로운 클라이언트의 소켓을 큐에 넣어줌 +Queue_Socket_Pop() : 큐에 들어있는 소켓 하나를 반환함 +Queue_isEmpty() : 큐가 비어있는지 알려줌 +Map_ContainKey(int key) : 해시맵에 키가 존재하는지 알려줌 +Map_getValue(int key) : 해시맵에 있는 해당 키의 스레드를 반환함 +Map_getkeySet() : 해시맵에 있는 키들을 반환함 +Map_put(id:int, thread:Client_Thread) : 해시맵에 해당 id와 스레드를 넣어줌 +get_Clientnum() 클라이언트의 수를 반환함 +getMonitorOos() : 옵저버에게 데이터를 보내는 변수를 반환함 +new_Client_access(ID:byte) : 새로운 클라이언트가 접속했음을 옵저버에게 로그인 패킷을 보내며 알리고, DB를 세팅 +addDB(RTLS_ID:int, RTLS_X:int, RTLS_Y:int, RTLS_STATE:byte) +Remove_Client(ID:int) : 해당 클라이언트의 Client_Thread를 종료하고, client_map에서 지운 후 db의 Remove_Client()를 호출한 후 옵저버에게 클라이언트가 종료했음을 알림

2.2.3. accepter

Attributes
-Server:RTLS_Server : RTLS_Server의 변수들을 사용하기 위한 변수
Methods
+accepter(server:RTLS_Server) : 생성자에서 Server 변수를 설정 +void run() : 새로운 클라이언트가 접속하는 것을 대기하며 접속하면 큐에 소켓을 넣어줌

2.2.4. worker

Attributes
-Server:RTLS_Server : RTLS_Server의 변수들을 사용하기 위한 변수
Methods
+worker(server:RTLS_Server) : 생성자에서 Server 변수를 설정 +void run() : 큐에 새로운 클라이언트의 소켓이 들어올 때마다 Client_Thread를 할당해줌

2.2.5. Client_Thread

Attributes
-Id:int : 클라이언트의 ID를 저장하는 변수 -RTLS_STATE:int = 0 : 클라이언트의 현재 상태를 저장하는 변수 -RTLS_X:int = 0 : 클라이언트의 현재 x위치를 저장하는 변수 -RTLS_Y:int = 0 : 클라이언트의 현재 y위치를 저장하는 변수 -socket:Socket : 클라이언트의 소켓을 저장하는 변수 -is:InputStream = null; : 클라이언트에게 데이터를 받을 때 사용할 변수 -os:OutputStream : 클라이언트에게 데이터를 전송할 때 사용할 변수 -ois:ObjectInputStream : 클라이언트에게 데이터를 받을 때 사용할 변수 -oos:ObjectOutputStream : 클라이언트에게 데이터를 전송할 때 사용할 변수 -buf:byte[] = new byte[512] : 클라이언트한테 받은 데이터를 저장하는 변수 -Server:RTLS_Server : RTLS_Server의 변수들을 사용하기 위한 변수
Methods
+Client_Thread(socket1:Socket , server:RTLS_Server, monitor_thread:monitor_thread) : 생성자에서 socket, server, monitor_thread를 설정하며, is, ois, os, oos를 설정한다. 이후 클라이언트의 ID를 할당하며 할당된 ID는 클라이언트에게 로그인 패킷을 보내며 알린다. 이후 서버의 해시맵과 DB를 설정한다. +getOos() : oos를 반환함 +get_RTLS_STATE() : 클라이언트의 현재 상태를 반환함 +get_RTLS_X() : 클라이언트의 현재 x위치를 반환함 +get_RTLS_Y() : 클라이언트의 현재 y위치를 반환함 +run() : 클라이언트에게 데이터가 올때까지 대기하며 데이터가 오면 해당되는 데이터에 맞게 처리함

2.2.6. monitor_thread

Attributes
−os_monitor:OutputStream : 옵저버에게 데이터를 전송할 때 사용할 변수 −oos_monitor:ObjectOutputStream : 옵저버에게 데이터를 전송할 때 사용할 변수 −is_monitor:InputStream = null : 옵저버에게 데이터를 받을 때 사용할 변수 −ois_monitor:ObjectInputStream : 옵저버에게 데이터를 받을 때 사용할 변수 −Server:RTLS_Server : RTLS_Server의 변수들을 사용하기 위한 변수 −monitor_receiver:monitor_receiver : 옵저버에서 오는 데이터를 받고 처리하는 스레드
Methods
+monitor_thread(server:RTLS_Server) : 생성자에서 server 변수를 설정하고 옵저버의 접속을 대기하고 접속하면 해당 소켓을 통해 os_monitor, oos_monitor, is_monitor, ois_monitor을 설정한 후 monitor_receiver을 생성하여 실행함 +run() : 1초마다 모든 클라이언트의 상태와 위치 데이터를 만들어 전송함 +ObjectOutputStream getOos() : oos_monitor를 반환함

2.2.7. monitor_receiver

Attributes
−oos_monitor:ObjectOutputStream : 옵저버에게 데이터를 전송할 때 사용할 변수 −ois_monitor:ObjectInputStream : 옵저버에게 데이터를 받을 때 사용할 변수 −server:RTLS_Server : RTLS_Server의 변수들을 사용하기 위한 변수 −buf:byte[] = new byte[512] : 옵저버에게 받은 데이터를 저장하는 변수
Methods
+monitor_receiver(oos_monitor:ObjectOutputStream, ois_monitor:ObjectInputStream, server:RTLS_Server) : 생성자에서 server, ois_monitor, oos_monitor 변수를 설정함 +run() : 옵저버에게 데이터가 올 때까지 대기하며 데이터가 오면 해당되는 데이터에 맞게 처리함

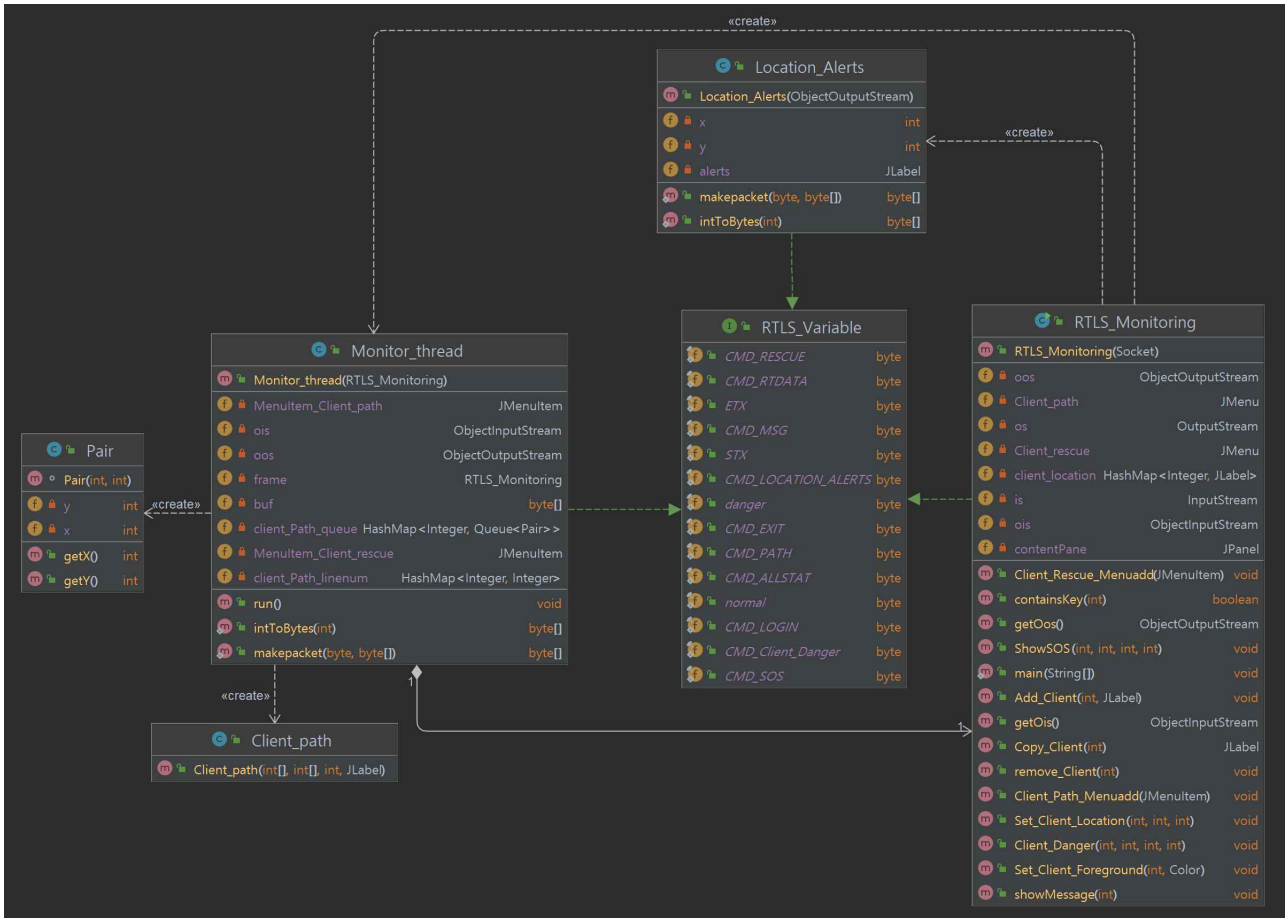
2.2.8. DB

Attributes
−db:HashMap<Integer, ArrayList<Client_Info>> : 클라이언트들의 이전 위치와 상태 정보들을 각 클라이언트별로 저장하고 관리하는 해시맵
Methods
+DB() : 생성자에서 db 변수를 설정함 +getClient(ID:int) : db에 있는 해당 ID의 ArrayList를 반환함 +addDB(ID:int, x:int , y:int, state:byte) : 해당 ID의 ArrayList에 위치와 상태정보를 추가하며, ArrayList의 크기가 100이 넘으면 index가 짝수인 값들은 모두 제거함 +newClient(ID:int) : db에 해당 ID와 ArrayList를 생성하여 해시맵에 추가함 +Remove_Client(ID:int) : 해당 클라이언트를 db에서 지움

2.2.9. Client_Info

Attributes
−x:int : 클라이언트의 x위치를 저장하는 변수 −y:int : 클라이언트의 y위치를 저장하는 변수 −state:byte : 클라이언트의 상태를 저장하는 변수
Methods
+Client_Info(x:int, y:int, state:byte) : 생성자에서 x, y, state를 설정함 +getState() : state를 반환함 +getX() : x를 반환함 +getY() : y를 반환함

2.3. Observer



<그림 4> Observer Class Diagram

2.3.1. RTLS_Variable

Attributes
+STX:byte = 0x02 : 데이터의 시작을 의미하는 값 +ETX:byte = 0x03 : 데이터의 끝을 의미하는 값 +CMD_RTDATA:byte = 0x00 : 클라이언트의 위치와 상태 데이터를 의미하는 값 +CMD_ALLSTAT:byte = 0x01 : 모든 클라이언트의 상태 데이터를 의미하는 값 +CMD_MSG:byte = 0x02 : 메시지 데이터를 의미하는 값 +CMD_LOGIN:byte = 0x10 : 새로운 클라이언트 로그인 데이터를 의미하는 값 +CMD_Client_Danger:byte = 0x11 : 클라이언트 위험 알림 데이터를 의미하는 값 +CMD_SOS:byte = 0x33 : SOS 요청 데이터를 의미하는 값 +CMD_RESCUE:byte = 0x34 : 구조요청 데이터를 의미하는 값 +CMD_LOCATION_ALERTS:byte = 0x35 : 위치 경고 데이터를 의미하는 값 +CMD_EXIT:byte = 0x44 : 클라이언트 종료 데이터를 의미하는 값 +normal:byte = 0x00 : 클라이언트의 상태가 정상이라는 것을 의미하는 값 +danger:byte = 0xFF : 클라이언트의 상태가 위험하다는 것을 의미하는 값
Methods

2.3.2. RTLS_Monitoring

Attributes
−contentPane:JPanel : 옵저버 화면 패널 −client_location:HashMap<Integer,JLabel> = new HashMap<Integer,JLabel>() : 클라이언트들의 위치와 상태를 표시할 JLabel을 관리하는 해시맵 −Client_path:JMenu : 클라이언트의 이동경로를 확인하는 메뉴 −Client_rescue:JMenu : 클라이언트 구조를 요청하는 메뉴 −is:InputStream = null : 서버에게 데이터를 받을 때 사용할 변수 −ois:ObjectInputStream : 서버에게 데이터를 받을 때 사용할 변수 −os:OutputStream : 서버에게 데이터를 전송할 때 사용할 변수 −oos:ObjectOutputStream : 서버에게 데이터를 전송할 때 사용할 변수
Methods
+RTLS_Monitoring() : 생성자에서 옵저버 화면을 구성함 +getOos() : oos를 반환함 +getOis() : ois를 반환함 +containsKey(id:int) : 해시맵에 해당 ID가 있는지 알려줌 +Set_Client_Location(id:int, x:int, y:int) : 해당 ID의 JLabel의 위치를 조정함 +Set_Client_Foreground(id:int, color:Color) : 해당 ID의 JLabel의 색(상태)을 조정함 +Add_Client(id:int, label:JLabel) : 해시맵과 화면 패널에 해당 ID로 JLabel을 넣음 +remove_Client(id:int) : 해당 클라이언트의 Client_path, Client_rescue, client_location, 화면패널에서 삭제함 +Copy_Client(id:int) : 해당 ID의 JLabel을 복사하여 반환함 +Client_Path_Menuadd (menu_item:JMenuItem) : Client_path 메뉴에 새로운 클라이언트를 추가함 +Client_Rescue_Menuadd (menu_item:JMenuItem) : Client_rescue메뉴에 새로운 클라이언트를 추가함 +showMessage(id:int) : 해당 ID의 클라이언트가 접속했음을 알림 +Client_Danger(id:int, int x, int y, int state) : 클라이언트가 위험상황임을 알림 +ShowSOS(ID:int, X:int, Y:int, State:int) : 클라이언트의 SOS요청 알림을 띄움 +Client_Exit(ID:int) : 해당 클라이언트의 JLabel을 화면 패널에서 제거하고 해시맵에서 지움

2.3.3. Monitor_thread

Attributes
−ois:ObjectInputStream : 서버에게 데이터를 받을 때 사용할 변수 −oos:ObjectOutputStream : 서버에게 데이터를 전송할 때 사용할 변수 −frame:RTLS_Monitoring : 옵저버의 화면 패널의 변수를 사용하기위한 변수 −buf:byte[] = new byte[512] : 서버에서 온 데이터를 저장하는 변수 −client_Path_linenum:HashMap<Integer, Integer> = new HashMap<Integer, Integer>() : 클라이언트의 이동경로를 그릴 때 그릴 선을 저장하는 해시맵 −client_Path_queue:HashMap<Integer, Queue<Pair>> = new HashMap<Integer, Queue<Pair>>() : 클라이언트의 이동경로를 그릴 때 이동경로를 저장하는 해시맵 −MenuItem_Client_path:JMenuItem : 클라이언트의 이동경로를 띄워주는 메뉴 −MenuItem_Client_rescue:JMenuItem : 클라이언트 구조 요청을 하는 메뉴
Methods
+Monitor_thread(frame:RTLS_Monitoring) : 생성자에서 frame, ois, oos 변수를 설정함 +run() : 서버에게 데이터가 올 때까지 대기하며 데이터가 오면 해당되는 데이터에 맞게 처리함

2.3.4. Client_path

Attributes
Methods
+Client_path(X_Array:int[], Y_Array:int[], Line_Num:int, id:JLabel) : 이동경로와 현재 클라이언트의 JLabel을 통해 새로운 화면에서 이동경로를 띄움

2.3.5. Pair

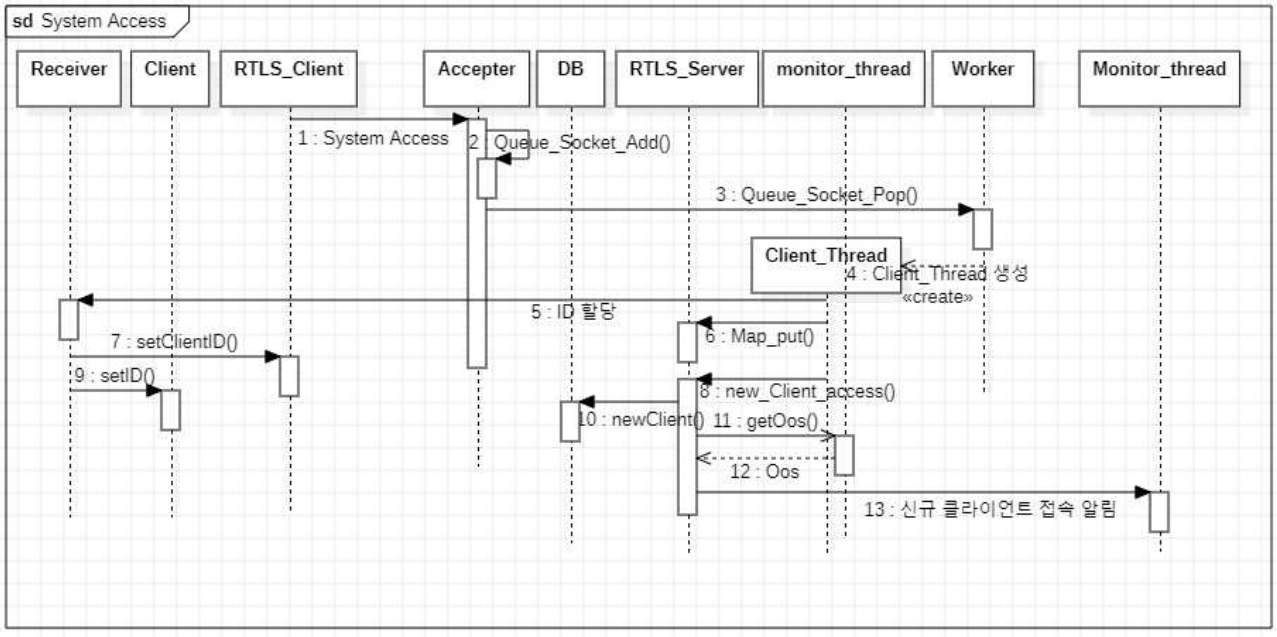
Attributes
−x:int : x위치를 저장하는 변수 −y:int : y위치를 저장하는 변수
Methods
+Pair(x:int, y:int) : 생성자에서 x,y를 설정함 +getX() : x를 반환함 +getY() : y를 반환함

2.3.6. Location_Alerts

Attributes
−x:int : 위험 지역의 x위치를 저장하는 변수 −y:int : 위험 지역의 y위치를 저장하는 변수
Methods
+Location_Alerts(oos:ObjectOutputStream) : Location_Alerts의 화면을 구성하고, oos를 설정함

3. Sequence diagram

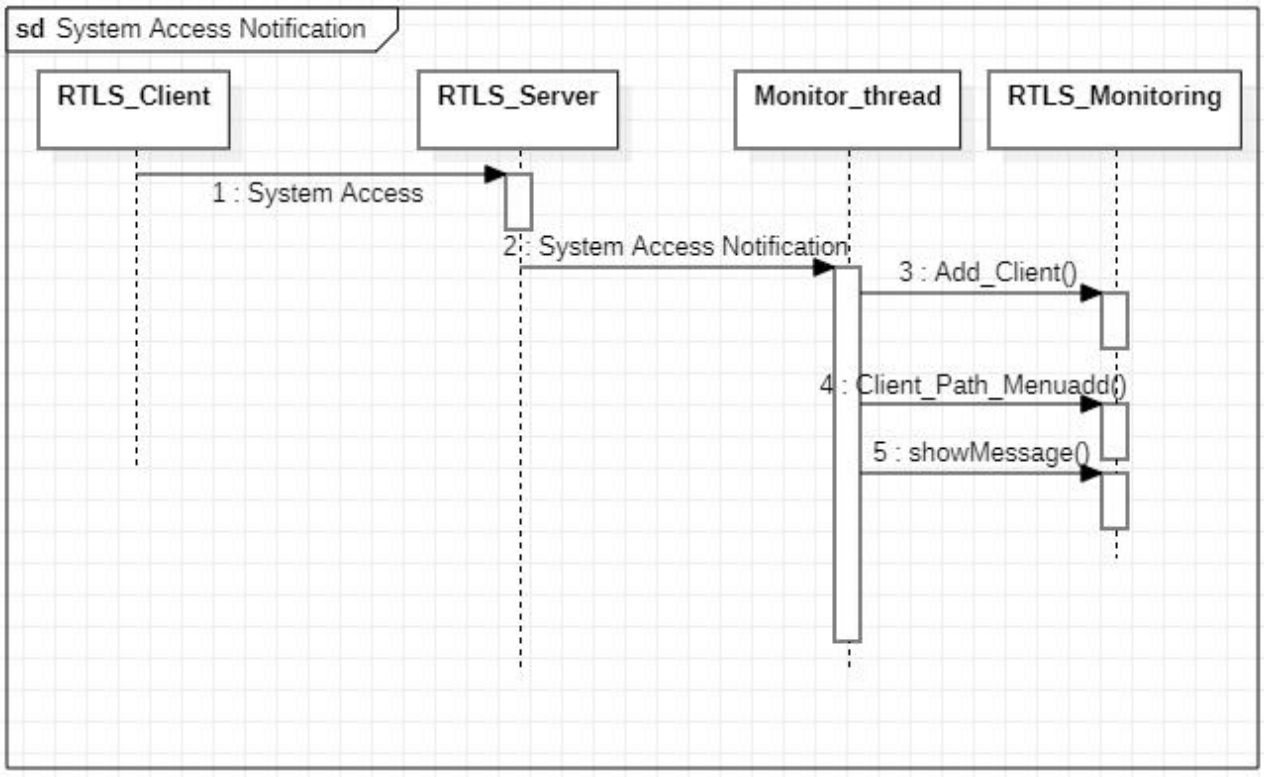
3.1. System Access



<그림 5> System Access Sequence diagram

위 그림은 클라이언트가 System Access하는 과정이다. System Access하는 순간 Accepter은 큐에 클라이언트 소켓을 넣은 후 Worker가 이를 이어받아 Client_Thread를 할당한다. Client_Thread가 생성되었을 때 Client_Thread에서 클라이언트에게 ID를 할당해준다. 그 후 Client_Thread에서는 서버에 새로운 클라이언트가 접속했음을 알림과 동시에 해시맵, DB설정, 옵저버에게 알림을 진행한다. 그동안 ID를 할당받은 클라이언트는 Client 클래스와 RTLS_Client의 ID를 설정해준다.

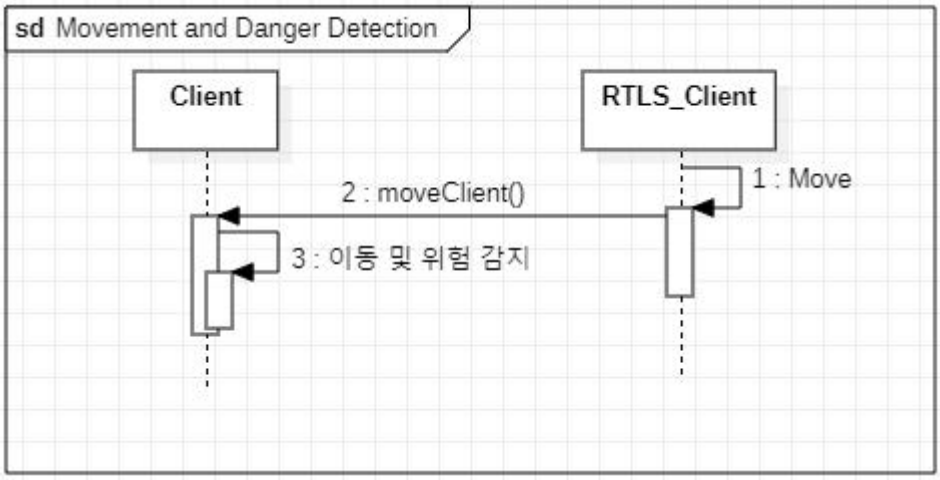
3.2. System Access Notification



<그림 6> System Access Notification Sequence diagram

위 그림은 RTLS에서 System Access Notification을 하는 과정이다. 클라이언트가 System Access를 하면 옵저버에게 System Access Notification을 한다. 이때 옵저버는 Add_Client()를 실행하여 새로운 클라이언트의 JLabel을 화면 패널에 넣고 이후 해시맵에 관리할수있게 설정해준다. 이후 이동경로를 확인할수있게 Client_Path_Menuadd()를 통해 메뉴에도 클라이언트를 할당해주며 마지막으로 옵저버에게 새로운 클라이언트가 왔음을 showMessage()를 통해 알린다.

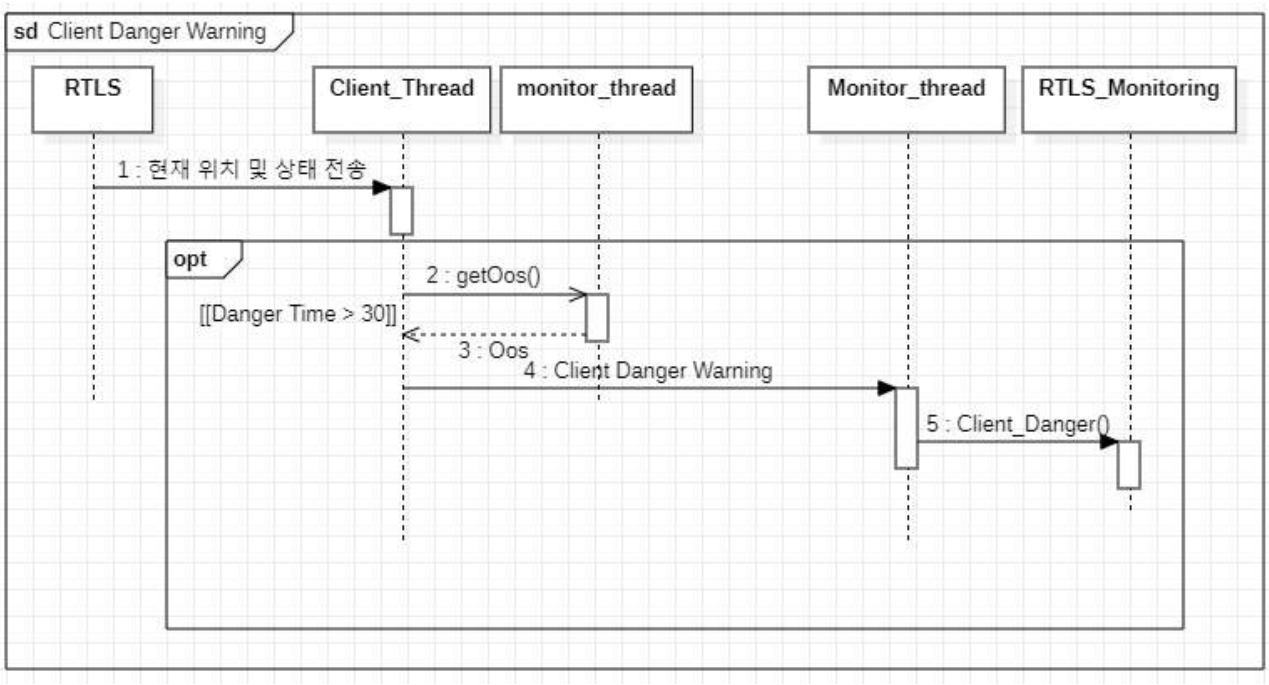
3.3. Movement and Danger Detection



<그림 7> Movement and Danger Detection Sequence diagram

위 그림은 클라이언트가 Movement and Danger Detection을 하는 과정이다. 클라이언트가 키보드를 통해 움직이면 Client로 moveClient()를 통해 클라이언트를 이동한다. 이때 Client는 JLabel의 위치를 이동하고 위험지역인지 확인 후 색(상태)을 바꾸는 과정을 거친다.

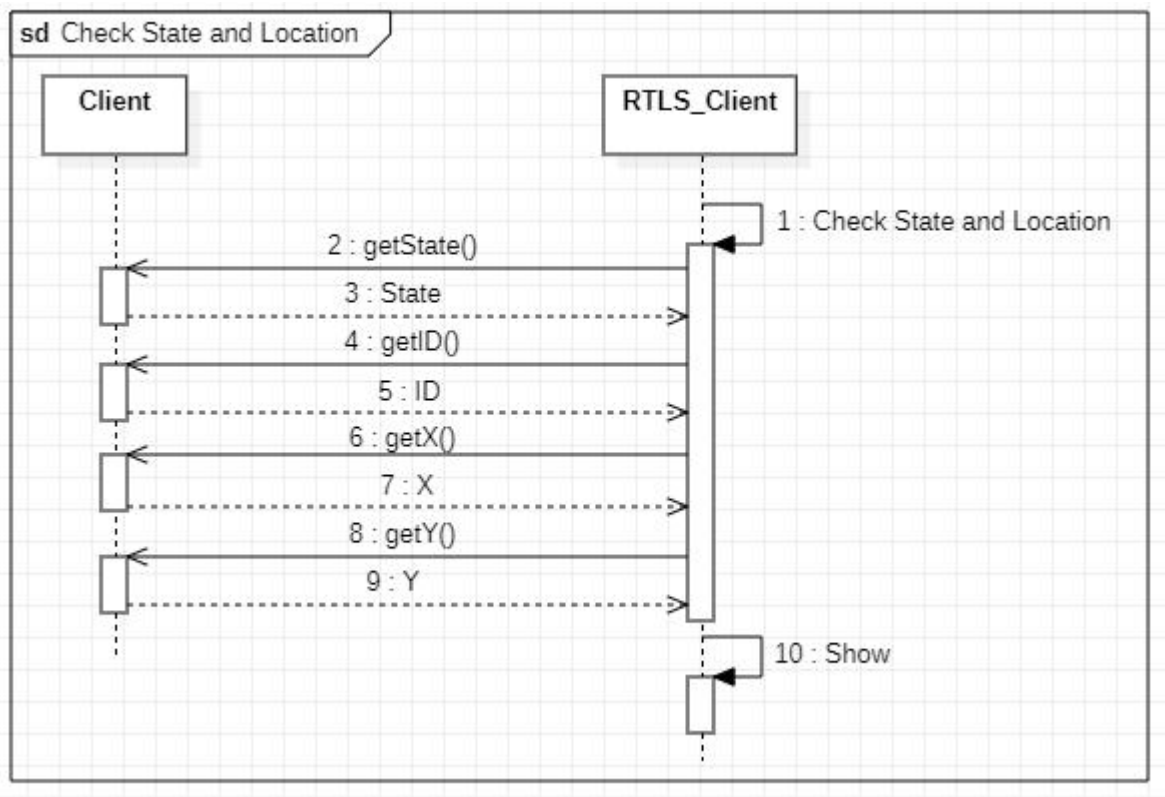
3.4. Client Danger Warning



<그림 8> Client Danger Warning Sequence diagram

위 그림은 Client Danger Warning의 과정이다. 클라이언트가 RTLS에서 1초마다 현재 위치 및 상태를 전송할 때 30번 넘게 위험지역에 머물 경우 클라이언트가 위험상황에 빠졌다고 가정하고 옵저버에게 Client Danger Warning을 한다. 이때 Client Danger Warning 데이터를 받은 Monitor_thread는 Client_Danger()을 통해 옵저버에게 위험 상황을 알린다. 알린 후 지속적으로 클라이언트가 위험지역에 머물 경우 15초 마다 한번씩 지속적으로 Client Danger Warning을 한다.

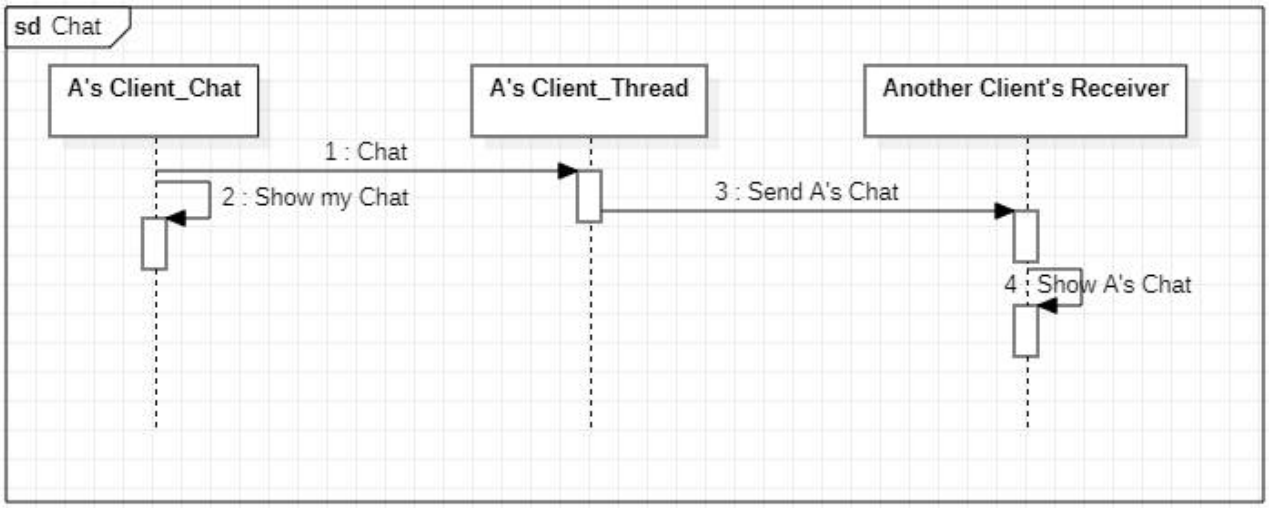
3.5. Check State and Location



<그림 9> Check State and Location Sequence diagram

위 그림은 클라이언트가 Check State and Location을 하는 과정이다. 클라이언트가 Check State and Location을 요청하면 Client 클래스에게 State, ID, X, Y를 얻은 후 클라이언트에게 메시지를 보여주며 Check State and Location이 종료되게 된다.

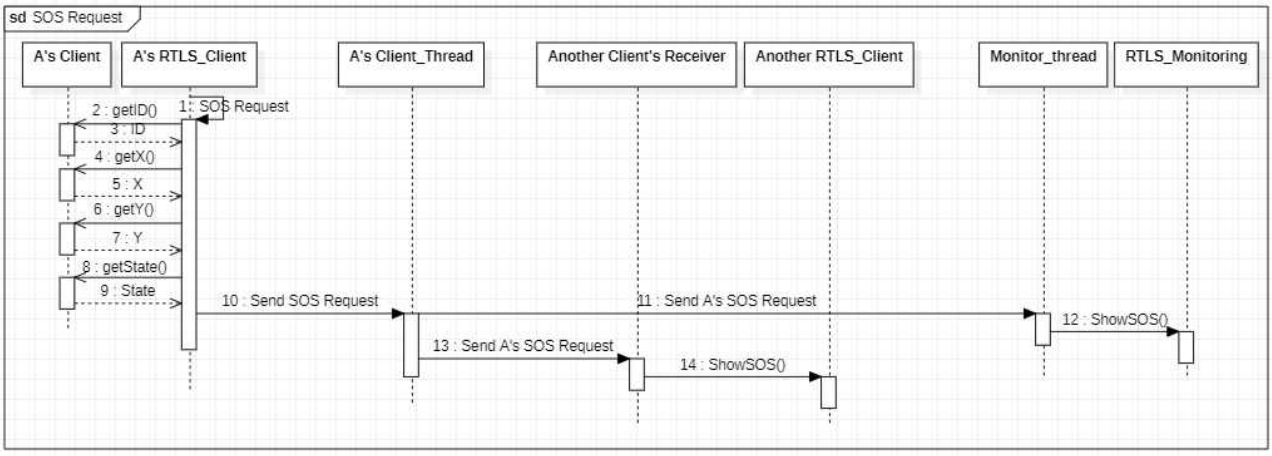
3.6. Chat



<그림 10> Chat Sequence diagram

위 그림은 클라이언트 A가 채널을 통해 Chat을 하는 과정이다. 채널을 통해 채팅을 보내면 A에게 할당된 Client_Thread는 A를 제외한 모든 클라이언트에게 해당 메시지를 전달한다. 전달받은 클라이언트들은 해당 채널에 메시지를 보여주며 끝난다.

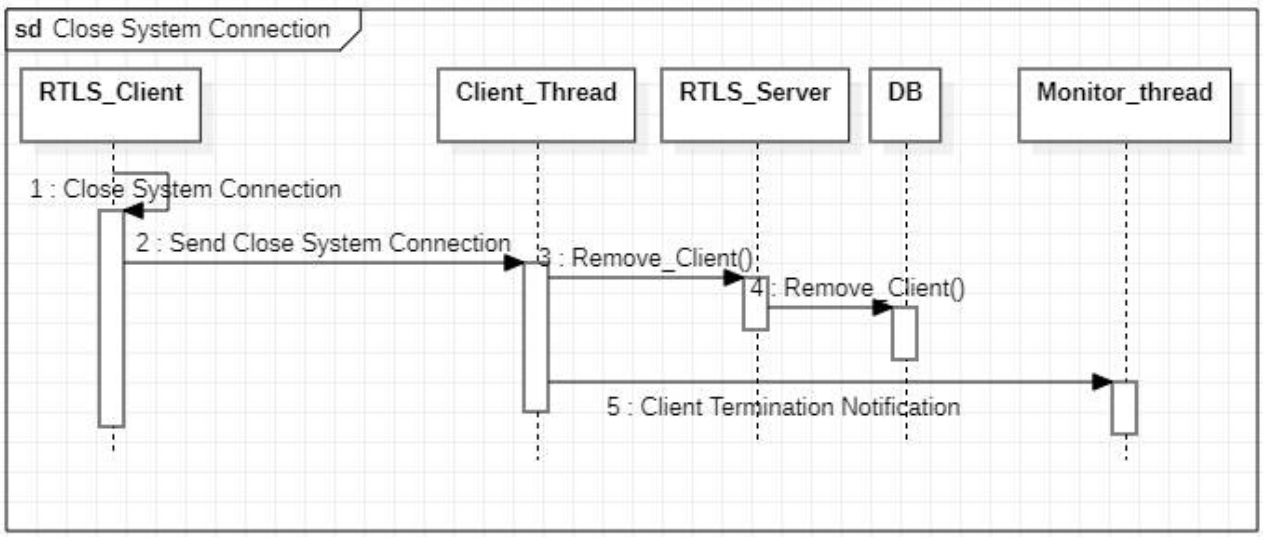
3.7. SOS Request



<그림 11> SOS Request Sequence diagram

위 그림은 클라이언트 A가 SOS Request를 하는 과정이다. A가 SOS Request를 할 때 A의 Client 클래스에서 ID, X, Y, State를 얻어온 후 SOS Request를 한다. 이때 A의 Client_Thread는 접속해있는 모든 사용자들에게 해당 메시지를 전달하며 각 사용자들은 ShowSOS()를 통해 A 클라이언트의 SOS Request를 보여주며 끝난다.

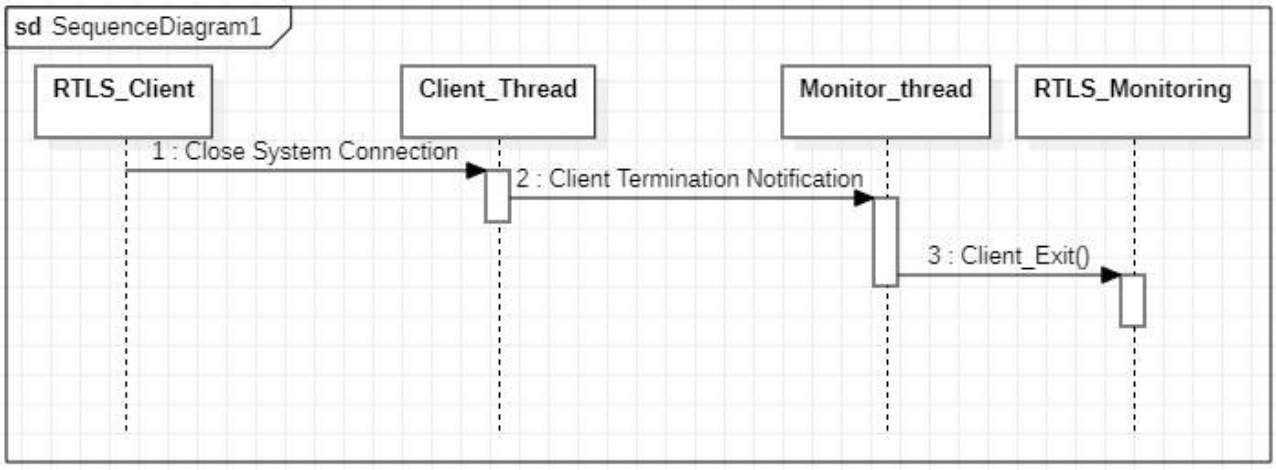
3.8. Close System Connection



<그림 12> Close System Connection Sequence diagram

위 그림은 클라이언트가 Close System Connection하는 과정이다. 클라이언트가 Close System Connection을 하면 서버로 해당 메시지를 전송한다. 이 메시지를 받은 서버는 해당 클라이언트에게 할당된 Client_Thread, 해시맵, DB 등을 모두 회수한다. 이후 옵저버에게 Client Termination Notification을 보내며 끝난다.

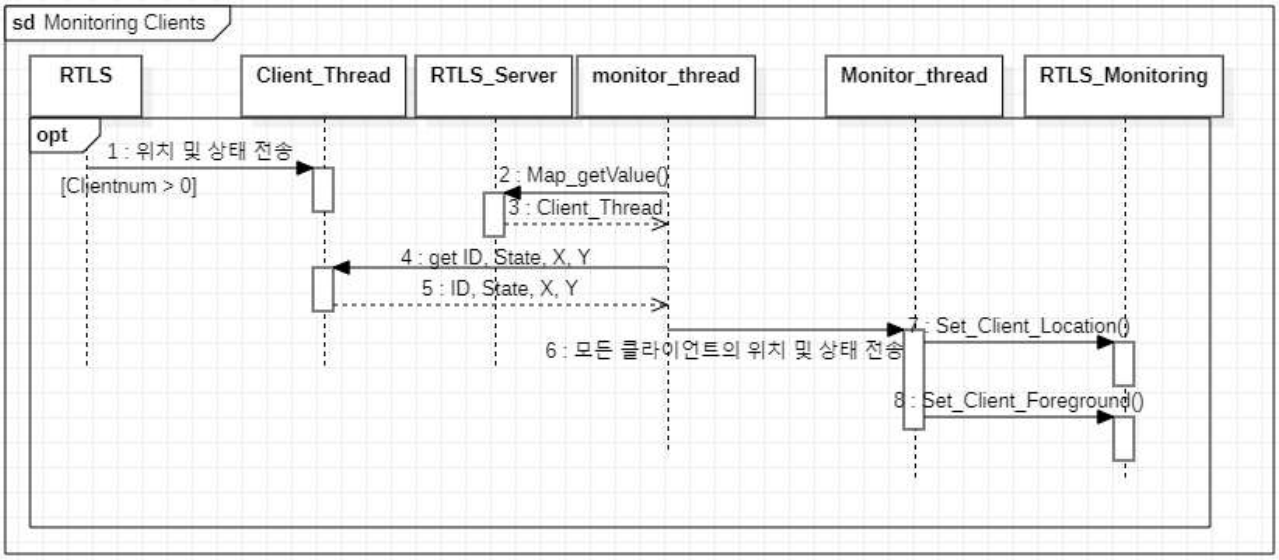
3.9. Client Termination Notification



<그림 13> Client Termination Notification Sequence diagram

위 그림은 서버에서 옵저버에게 Client Termination Notification하는 과정이다. 클라이언트가 Close System Connection을 하면 서버는 옵저버에게 Client Termination Notification을 한다. 옵저버에서 이제 해당 클라이언트를 모니터링 할 수 없게 해야하므로 Client_Exit()을 통해 해당 클라이언트의 JLabel과 해시맵에서 제거해준다.

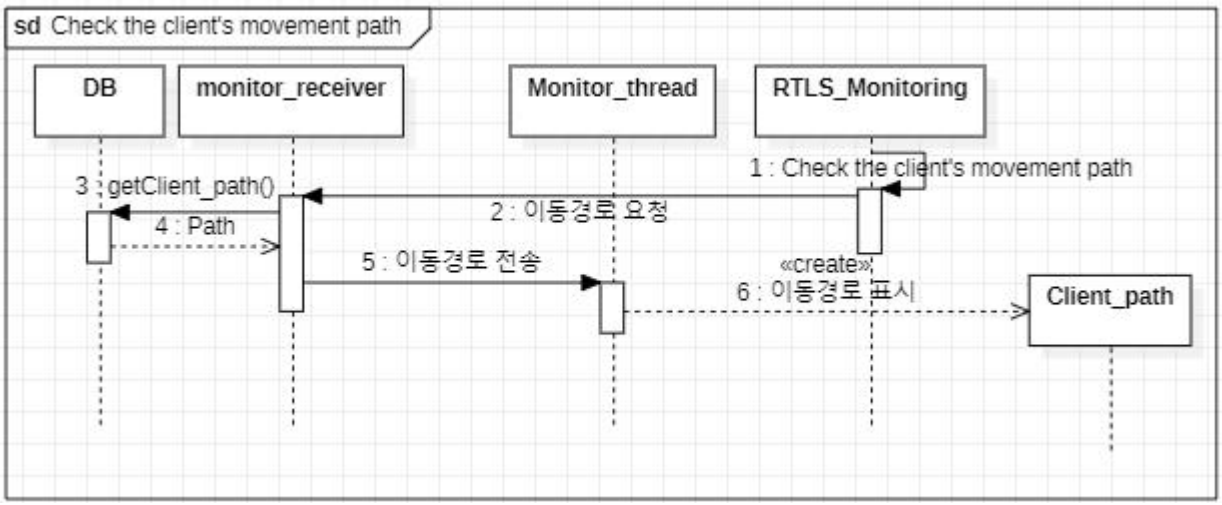
3.10. Monitoring Clients



<그림 14> Monitoring Clients Sequence diagram

위 그림은 옵저버가 Monitoring Clients을 하는 과정이다. 각 클라이언트의 RTLS 클래스에서 1초마다 서버로 현재 위치와 상태를 전송한다. Client_Thread에서 이를 저장해두며 서버의 monitor_thread는 1초마다 접속한 각 클라이언트의 Client_Thread를 얻어서 현재 위치 및 상태를 얻은 후 이를 종합하여 옵저버에게 전송한다. 전송받은 옵저버는 해당 데이터를 통하여 클라이언트들의 위치 및 상태를 변경하며 끝난다.

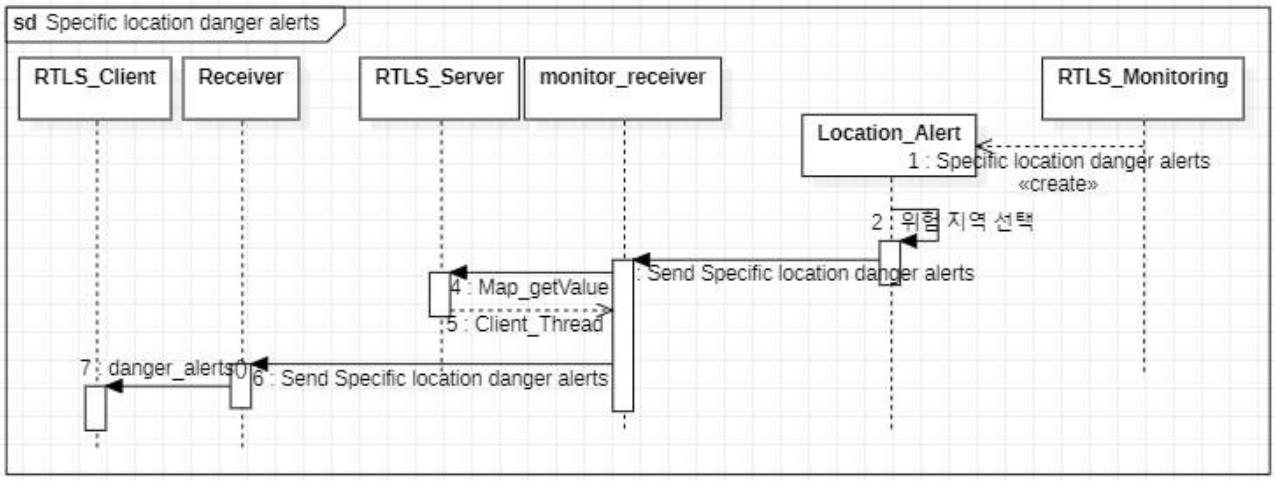
3.11. Check the client's movement path



<그림 15> Check the client's movement path Sequence diagram

위 그림은 옵저버가 Check the client's movement path를 하는 과정이다. 옵저버가 Check the client's movement path을 하면 서버로 해당 클라이언트의 이동경로를 요청한다. 요청 받은 서버는 DB로 getClient_path()를 통해 저장되어있는 이동경로를 받은 후 옵저버에게 전송한다. 전송을 모두 받은 옵저버는 Client_path 클래스를 통해 새로운 패널에 이동경로를 표시하며 끝난다.

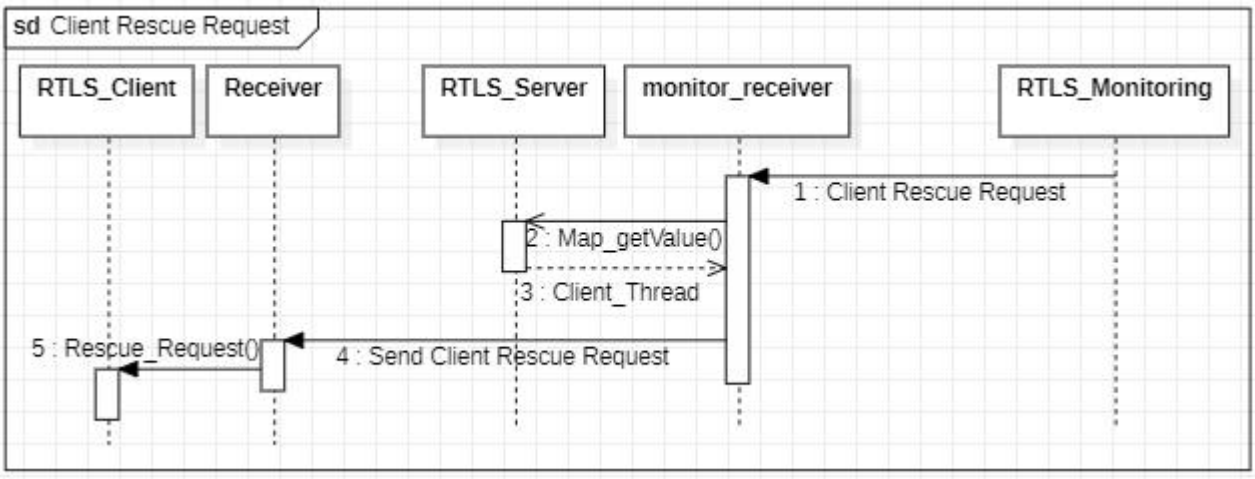
3.12. Specific location danger alerts



<그림 16> Specific location danger alerts Sequence diagram

위 그림은 옵저버에서 Specific location danger alerts을 하는 과정이다. 옵저버가 Specific location danger alerts를 할 때 Location_Alert 클래스를 생성하여 해당 패널에서 위험지역을 선택한다. 선택 후 서버로 전송하면 monitor_receiver는 각 클라이언트의 Client_Thread을 Map_getValue()를 통해 얻은 후 각 클라이언트에게 Specific location danger alerts를 전송한다. Specific location danger alerts을 받은 클라이언트는 danger_alerts()를 통해 해당 위치가 위험함을 알리며 끝난다.

3.13. Client Rescue Request

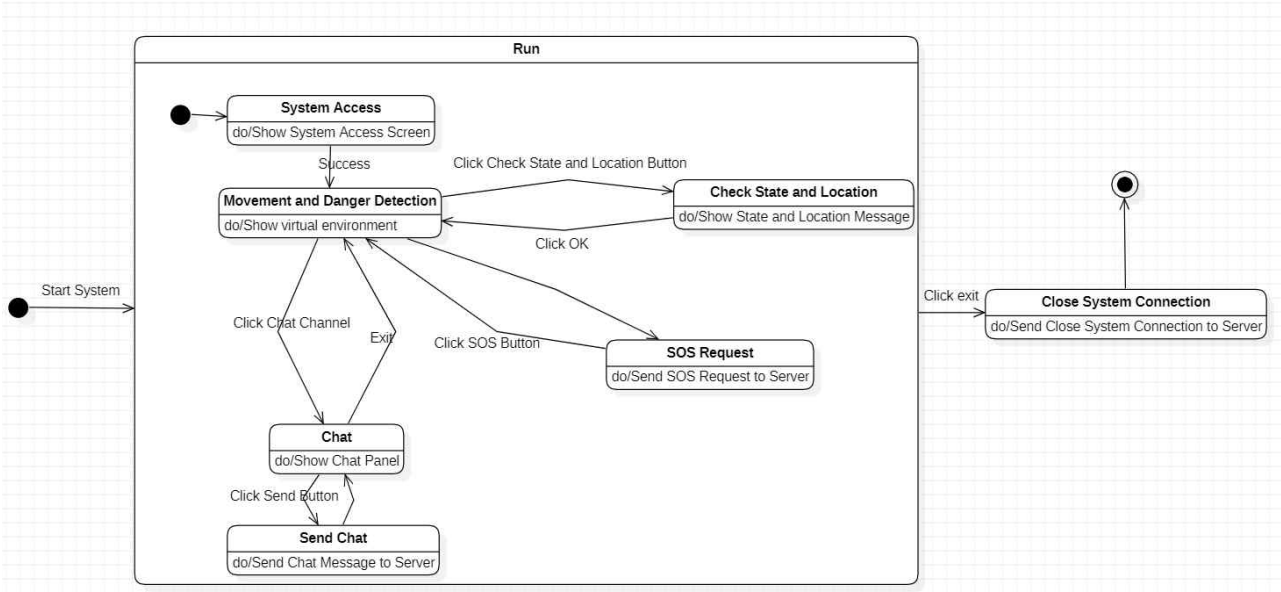


<그림 17> Client Rescue Request Sequence diagram

위 그림은 옵저버가 Client Rescue Request를 하는 과정이다. 구조요청을 보내려는 클라이언트의 JLabel을 통해 ID, X, Y, State를 얻은 후 서버로 Client Rescue Request를 요청한다. 이후 monitor_receiver는 Map_getValue()를 통해 각 클라이언트의 Client_Thread를 얻는다. 그 후 각 클라이언트에게 Client Rescue Request를 보내며 Client Rescue Request를 받은 클라이언트는 Rescue_Requese()를 통해 옵저버의 Client Rescue Request를 띄우며 끝난다.

4. State machine diagram

4.1. Client



<그림 18> Client State machine diagram

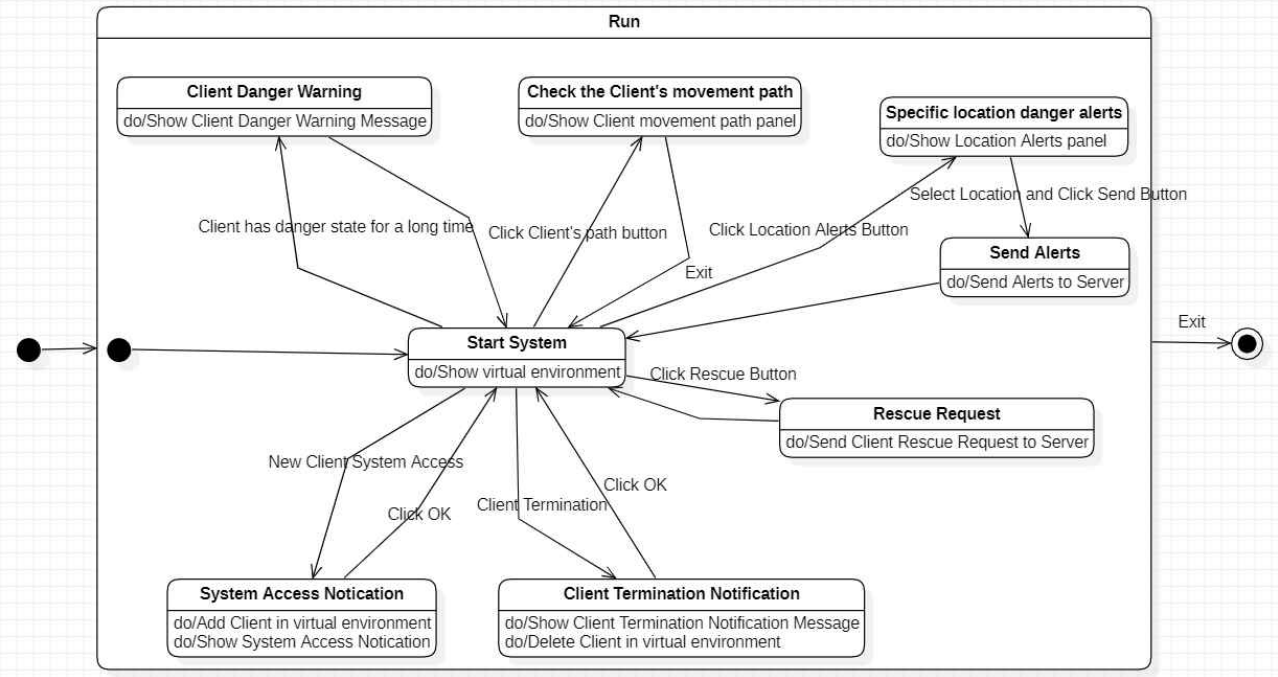
클라이언트가 시스템을 실행하면 시스템 접속 화면이 보인다. 접속을 시도하면 서버에서 ID를 할당해주고 접속에 성공한다. 이후 클라이언트는 가상 환경이 보여지고, 여기서 키보드를 통해 움직일 수 있다. 이때 현재 상태와 위치를 확인, 채팅, SOS 요청을 할 수 있다.

현재 상태와 위치를 확인하는 버튼을 누르면 현재 화면에서 자신의 X,Y 좌표의 위치와 위험 여부 상태를 알려주는 메시지 창이 보여진다. 확인 버튼을 누르면 다시 가상 환경 화면으로 돌아가게 된다.

SOS 요청 버튼을 누르면 서버로 SOS 요청이 전달되게 되고 이때 시스템에 접속된 모두에게 SOS 요청이 전달되어 알림이 가게 된다. 서버로 SOS 요청이 전달되면 다시 가상 환경 화면으로 돌아가게 된다.

그리고 채팅 채널을 누르게 되면 새로운 채팅 화면 패널이 생성되며 여기서 메시지를 입력하여 Send 버튼을 누르면 서버로 채팅 메시지가 전달되며 해당 채널로 메시지가 전달된다. 채팅 화면에서 X버튼을 눌러 나가게되면 채팅 화면 패널을 사라지게된다. 클라이언트의 가상 환경 화면에서 X버튼을 누르면 서버로 해당 클라이언트의 시스템 접속 종료 알림이 전달되며 이후 종료된다.

4.2. Observer



<그림 19> Observer State machine diagram

옵저버가 시스템을 시작하면 클라이언트와 같은 가상 환경이 보여진다. 새로운 클라이언트가 접속하면 옵저버에게 알림이 가며 가상환경에 클라이언트가 추가되어 실시간으로 모니터링 할 수 있게 된다. 클라이언트가 시스템에 접속을 종료하면 접속 종료 알림이 옵저버에게 전달되며, 가상환경에서 해당 클라이언트를 제거한다.

특정 클라이언트에 대한 구조 요청 버튼을 누르면 서버로 해당 클라이언트의 구조요청 메시지가 전달되며 메시지는 해당 클라이언트와 다른 클라이언트들에게 전달된다.

특정 위치 위험 알림 버튼을 누르게되면, 가상환경이 포함된 패널이 보여지게된다. 여기서 위치를 마우스로 선택하여 Send 버튼을 누르면 창이 닫히며 서버로 해당 위치에 대한 경고 메시지를 모든 클라이언트들에게 전송한다.

특정 클라이언트의 이동경로 버튼을 누르면 가상환경이 포함된 새로운 이동경로 패널 창이 보여지게된다. X버튼을 눌러 패널을 닫으면 이동경로 패널 창이 닫히게된다.

클라이언트가 위험상태로 오랫동안 지속되면 옵저버에게 해당 클라이언트의 ID, 위치, 상태가 포함된 메시지가 보여지게된다.

5. Implementation requirements

5.1. H/W platform requirements

- CPU : Intel i3+
- RAM : 4GB+
- HDD / SSD : 10GB+
- Network : 연결 필요

5.2. S/W platform requirements

- OS : Microsoft Window 7이상
- Implemetaion Language : JAVA

6. Glossary

Term	Description
Attribute	객체의 상태를 나타내며, 데이터 또는 속성이라고 함
Class Diagram	"Class"라고 하는 객체지향 설계단위를 이용하여 시스템의 정적인 structure를 표현한 다이어그램
Method	객체가 메시지를 받아 실행해야 할 객체의 구체적인 연산을 정의한 것, 전통적 시스템의 함수나 프로시저에 해당하는 연산 기능
Operation	객체의 행동(behavior) 및 다른 객체에 대하여 제공하는 것
Sequence Diagram	객체간의 동적 상호작용을 시간적 개념을 중심으로 모델링하는 과정
Sate Machine Diagram	객체 lifetime동안의 변환될 수 있는 모든 상태를 정의해둔 다이어그램

7. References

- 강의자료 : Structural Modeling II, Behavior Modeling I, II
- 참고자료 http://staruml.sourceforge.net/docs/user-guide(ko)/toc.html