**●Manager.h**

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <string.h>

#include <stdlib.h>

#include <stdio.h>

#include <unistd.h>

#include <pthread.h>

#include <arpa/inet.h>

#include <sys/stat.h>

**#include "LinkedList.h"**

#include <time.h>

#include "/usr/include/mysql/mysql.h"

#define BUF\_LEN 128

#define TEMP 1

#define HUMD 2

#define LUX 3

#define MOTION 4

#define TCP\_PORT 8080

#define SPRNG\_AUTUM 1

#define SUMMER 2

#define WINTER 3

#define DBHOST "DBHOST"

#define DBUSER "USER"

#define DBPASS "PASSWORD"

#define DBNAME "SmartHomeServer"

extern pthread\_mutex\_t mutexLock;

===========================================================================

**1. Manager Class**

---------------------------------------------------------------------------

[필드 정보/ 이름 / 역할]

private:

struct sockaddr\_in serverAddr, clientAddr

-스마트 센서들과 통신을 하기위한 소켓 주소

int on

-스마트 센서들과 통신을 하기위한 변수

int serverFd, clientFd

- 파일디스크립터 변수

socklen\_t len, lenOut

-소켓 길이를 담을 변수

int port

-통신 포트 번호

SensorManager\* sensorManager

-스마트 센서 관리를 위한 센서 매니저

char clientName[BUFSIZ]

-연결된 클라이언트 주소를 담을 변수

char myName[BUFSIZ]

-매니저 아이피

MYSQL connector

-sql서버와 연결할 커넥터

MYSQL\_RES\* sqlResult

-sql쿼리 결과를 담을 변수

MYSQL\_ROW sqlRow

char query[1024]

-쿼리문

int myCode

-sql 서버에 저장된 매니저 코드

char emptyTempMsg[BUF\_LEN + 1]

char emptyHumdMsg[BUF\_LEN + 1]

char emptyLuxMsg[BUF\_LEN + 1]

char emptyMotionMsg[BUF\_LEN + 1]

-해당 스마트 센서가 없을시 보낼 메시지

char bufIpz[BUFSIZ]

char buf[BUF\_LEN + 1]

-버퍼들

int weather

-처음 시작시 가져오는 월

float optimumTemp

float optimumHumd

float optimumLux

float optimumMotion

char optimumTempMsg[BUF\_LEN + 1]

char optimumHumdMsg[BUF\_LEN + 1]

char optimumLuxMsg[BUF\_LEN + 1]

char optimumMotionMsg[BUF\_LEN + 1]

-센서의 적정 값을 설정하고 주고 받기위한 메시지

---------------------------------------------------------------------------

SMARTSENSOR MANAGE FUNCTION(1-1 ~ 1-32)

**[메소드 번호./ 메소드 이름/ 정보/해당 메소드 내에서 사용한 메소드/역할]**

public:

1-1. Manager()

1-2. Manager(int port)

사용한 메소드 : 1-24 | 1-25

역할 : 초기화, 오늘의 날짜를 가져와 적정값 자동 설정

1-3. void ConnectSetting()

역할 : 센서관리를 위한 통신 준비 메소드

1-4. void SetOptMsg(char Msg[], int n)

역할 : 각 온도,습도, 조도별 적정값 설정위한 메시지 설정

=> 현재 적정값 설정은 각 기기당이 아닌 온도별 습도별로 가능. (예를 들어 온도기기가 두 개이면 각각의 적정값 설정 불가)

1-5. void\* Connect(void \*data)

사용한 메소드 : 1-27 | 1-23

역할 : 스마트 센서의 연결을 기다리며 스마트 기기의 요청에 따라 수행하 도록 함

1-6. static void\* CallBackConnect(void\* server\_ptr)

사용한 메소드 : 1-5

역할 : 스레드 사용을 위해 만든 static한 콜백 함수. 1-5를 호출한다.

---------------------------------------------------------------------------

GET, SET FUNCTION (1-7 ~ 1-22)

1-7. struct sockaddr\_in GetServerAddr()

1-8. struct sockaddr\_in GetClientAddr()

1-9. int GetServerFd()

1-10. int GetClientFd()

1-11. socklen\_t GetClientLen()

1-12. void SetClientLen(sockaddr\_in client)

1-13. int GetWeather()

1-14. float GetOptimumTemp()

1-15. float GetOptimumHumd()

1-16. float GetOptimumLux()

1-17. float GetOptimumMotion()

1-18. void SetWeather(int weather)

1-19. void SetOptimumTemp(float temp)

1-20. void SetOptimumHumd(float humd)

1-21. void SetOptimumLux(float lux)

1-22. void SetOptimumMotion(float motion)

---------------------------------------------------------------------------

1-23. void ShowMsg()

역할 : 적정값 전달 메시지 내용 출력

1-24. void SetOptimumDataWithWeather(int month)

사용한 메소드 : 1-18

역할 : 월을 받아 적정값을 설정

1-25. void SetOptimumDataToString()

역할 : float형의 적정값을 string형식으로 변환

1-26. void SetOptimumStringToData()

역할 : string형의 적정값을 float형으로 변환

1-27. void ManageMsg(char buf[])

사용한 메소드 : 1-29 | 1-30 | 1-31 | 1-32

역할 : 클라이언트로부터 받은 메시지를 메시지 요청에 따라 다른 함수 호출

1-28. char\* MsgCutForGetSendIpAddr(char buf[])

역할 : 특정기기의 센서 값을 요청했을 경우 메시지에 첨부한 기기의 아이피를 가 져옴

1-29. void ReceiveSensorMsg(char buf[])

역할 : 스마트 센서가 자신의 센서값을 보낼 경우 해당 센서의 정보와 센서데이터 저장

1-30. void SendSensorMsg(char buf[])

사용한 메소드 : 1-28

역할 : 어플에서 특정 기기의 센서 데이터 원할 경우 특정기기 정보가 없으면 없다 는 메시지를, 정보가 있다면 해당 데이터를 보내주는 메소드

1-31. void ReceiveOptimumMsg(char buf[])

사용한 메소드 : 1-26

역할 : 어플에서 적정값 설정시 메시지가 오면 해당 메시지를 읽고 적정값을 설정

1-32. void SendOptimumMsg(char buf[])

역할 : 매니저가 가지고 있는 적정값들을 각각의 적정값을 스마트 센서에게 알려줌

---------------------------------------------------------------------------

SERVER　COMMUNICATION FUNCTION(1-33 ~ 1-39)

1-33. void CommuniSetting()

역할 : 서버와 연결을위한 세팅 메소드

1-34. void SendCommuniMsg()

사용한 메소드 : 1-37 | 1-38

역할 : 시간당 센서 매니저가 가지고 있는 정보 서버에 전달

1-35. static void\* CallBackCommuniConnect(void\* server\_ptr)

사용한 메소드 : 1-34

역할 :1-34 메소드를 스레드로 사용하기 위한 콜백 메소드

1-36. void WriteQuery()

역할 : 필드에 있는 query를 쿼리해주는 메소드

1-37. void MangerSetQuery()

사용한 메소드 : 1-36

역할 : 서버에 자신에 대한 정보가 없으면 자신의 정보를 쿼리해주고 자신이 가진 센서기기가 각가 몇 개인지 쿼리해줌(예: 온도 2개 습도 1개 조도 0개 )

1-38. void WriteAllData(int hour)

사용한 메소드 : 3-15 | 1-36

역할 : 센서 매니저가 가지고 있는 데이터들 모두 쿼리

1-39. float MsgTransformToQuery(char msg[])

역할 : 센서 매니저는 데이터를 char배열 형식으로 가지고 있으므로 float형으로 변환 하기위한 함수

1-40. ~Manager()

**========================================================**

**●LinkedList.h**

#include <stdio.h>

#include <malloc.h>

#include <stdlib.h>

#include <limits.h>

#include <string.h>

#include<time.h>

#pragma warning (disable:4996)

#define FALSE 0

#define TRUE 1

**2. SmartThing Class**

---------------------------------------------------------------------------

public:

char\* ip

-스마트 센서의 ip

int sensorType

-스마트 센서 타입(예 조도, 습도,,온도)

char\* msg

-해당 기기가 가지고 있는 데이터 메시지

SmartThing\* link

-리스트 구현을 위한 포인터

int timer

-센서 오작동 감지를 위한 타이머 , 특정 시간이 지나도 데이터를 보내지아니면 오류로 판 단, 리스트에서 제거함 =>타이머 부분은 현재 잘못 구현해서 수정해야함!!

---------------------------------------------------------------------------

public:

2-1. SmartThing(char ip[], int sensorType, char msg[]);

역할 : 생성자와 필드 초기화

2-2. void setMsg(char buf[]);

역할 : 메시지를 저장하고 타이머 초기화와 시작을 함

2-3. ~SmartThing();

**===========================================================================**

**3. SensorManager Class**

---------------------------------------------------------------------------

private:

SmartThing \*\*phead

SmartThing \*head

int length

int tempLen

int humdLen

int luxLen

int motionLen

---------------------------------------------------------------------------

public:

3-1. SensorManager()

역할 : 연결리스트를 위한 초기화

3-2. void insertSmartThing(SmartThing \*p, SmartThing \*new\_node)

역할 : 선행노드앞에 새로운 노드 삽입

3-3. void removeSmartThing(SmartThing \*p, SmartThing \*removed)

역할 : 선행노드 다음의 노드 제거

3-4. int getLength()

역할 : 리스트 길이 반환

3-5. void error(char \*message)

역할 : 오류처리

3-6. void add(int position, SmartThing\* data)

사용한 메소드 : 3-15 | 3-2

3-15와 3-2를 사용하여 특정 위치에 SmartThing삽입

3-7. void addLast(SmartThing\* data)

사용한 메소드 : 3-4

역할 : 리스트 끝에 SmartThing삽입

3-8. void addFirst(SmartThing\* data)

역할 : 리스트 처음에 SmartThing삽입

3-9. int isEmpty()

역할 : 리스트가 비었는지 확인

3-10. void deleteSmartThing(int pos)

사용한 메소드 : 3-15 | 3-3

역할 : 특정위치 노드 제거

3-11. SmartThing\* getEntry(int pos)

사용한 메소드 : 3-15

역할 : 특정위치 노드 가져오기

3-12. void clear()

사용한 메소드 : 3-10

역할 : 전체 제거

3-13. void display()

역할 : 리스트안에 들어있는 정보 모두 출력

3-14. SmartThing \* isInList(int type, char ips[])

역할 : 스마트센서 타입과 ip로 센서 찾기

3-15. SmartThing \* getSmartThingAt(int pos)

역할 : 특정위치의 노드 반환

3-16. void AddTempLen()

3-17. void AddHumdLen()

3-18. void AddLuxLen()

3-19. void AddMotionLen()

역할 : 타입별로 스마트센서 개수 하나씩 증가

3-20. int GetTempLen()

3-21. int GetHumdLen()

3-22. int GetLuxLen()

3-23. int GetMotionLen()

역할 : 타입별로 스마트센서 길이 가져오기

3-24. void SubTempLen()

3-25. void SubHumdLen()

3-26. void SubLuxLen()

3-27. void SubMotionLen()

역할 : 타입별로 스마트 센서 개수 하나씩 감소

3-28. void SensorErrorCheck()

사용한 메소드 : 3-10

역할 : 타이머를 보고 특정 시간안에 메시지가 오지않으면 오류로 보고 삭제

=> 수정해야함

**===========================================================================**

**●Manager.h**

#ifndef TCP\_CLIENT\_H\_

#define TCP\_CLIENT\_H\_

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

#include <string.h>

#include <sys/socket.h>

#include <sys/stat.h>

#include <arpa/inet.h>

#define TCP\_PORT 5100

#define TEMP 1

#define HUMD 2

#define LUX 3

#define MOTION 4

===========================================================================

**4. Client Class**

---------------------------------------------------------------------------

[필드 정보/ 이름 / 역할]

private:

int sock

-홈매니저와 통신을위한 소켓

int on

-홈매니저와 통신을위한 변수

socklen\_t serverAddrLen

-홈 매니저와 통신을위한 서버소켓 길이

char ip[BUFSIZ]

-통신할 홈매니저의 아이피

struct sockaddr\_in serverAddr

-서버소켓 주소

char communicationOptimumMsgToSend[BUFSIZ]

-적정값을 받기위해 홈매니저에게 보내는 메시지

char communicationSensorMsgToSend[BUFSIZ]

-자신의 센서데이터를 보내기위해 홈매니저에게 보내는 메시지

char tempMsg[BUFSIZ + 1]

char humdMsg[BUFSIZ + 1]

char luxMsg[BUFSIZ + 1]

char motionMsg[BUFSIZ + 1]

-각 센서별 데이터를 담고있는 메시지

=========> 스마트 센서가 다른 스마트센서의 센서데이터를 모르도록 수정할 예정,,, 현재 ip주소도 필요하므로 서로 주고받더라도 이 부분은 수정해야함

---------------------------------------------------------------------------

**[메소드 번호./ 메소드 이름/ 정보/해당 메소드 내에서 사용한 메소드/역할]**

public:

4-1. Client()

사용한 메소드 : 4-21 | 4-19

역할 : ip설정과 메시지 초기화

---------------------------------------------------------------------------

GET,SET FUNCTION(4-2 ~ 4-19)

4-2. int GetSockAddr()

4-3. struct sockaddr\* GetServerAddr()

4-4. socklen\_t GetServerAddrLen()

4-5. char\* GetCommunicationSensorMsgToSend()

4-6. char\* GetCommunicationOptimumMsgToSend()

4-7. char\* GetTempMsg()

4-8. char\* GetHumdMsg()

4-9. char\* GetLuxMsg()

4-10. char\* GetMotionMsg()

4-11. void SetSockAddr(int)

4-12. void SetCommunicationOptimumMsgToSend(char[])

4-13. void SetCommunicationSensorMsgToSend(char[])

4-14. void SetTempMsg(char[])

4-15. void SetHumdMsg(char[])

4-16. void SetLuxMsg(char[])

4-17. void SetMotionMsg(char[])

4-18. char\* GetIp()

4-19. void SetIp(char [])

---------------------------------------------------------------------------

4-20. void ConnectionSetting(char\* argv)

-홈매니저와의 통신을위한 통신 세팅

4-21. void MsgSetting(char Data[], int n)

-센서별로 데이터문자열을 메시지에 저장

4-22. void PrintMsg()const

-각 센서 메시지 출력

4-23. ~Client();

**========================================================**

**●Sensor.h**

#include <stdio.h>

#include <string.h>

#include <errno.h>

#include <wiringPi.h>

#include <wiringPiSPI.h>

#include <time.h>

#define TYPE\_INIT 0

#define DATA\_INIT 0

#define VOLT\_INIT 0

#define PIN 7

#define PIR 4

===========================================================================

**5. Sensor Class**

---------------------------------------------------------------------------

[필드 정보/ 이름 / 역할]

private:

int sensorType

-센서 타입(온도인지,, 조도센서인지..)

int mcpData

-mcp3208을 통해 읽은 디지털 값

float volt

-읽은 디지털값을 볼트로 변환한 값

float realData

-실용값(온도라면 섭씨, 습도라면 %%..)

bool timerOn

-사물제어를 타이머 on off부울린

int electronicOnTime

-사물제어를 위한 타이머

clock\_t startTime

---------------------------------------------------------------------------

**[메소드 번호./ 메소드 이름/ 정보/해당 메소드 내에서 사용한 메소드/역할]**

public:

5-1. Sensor()

사용한 메소드 : 5-4 | 5-17

역할 : 필드값 초기화

5-2. int Detection()

사용한 메소드 : 5-22

역할 : mcp-3208을 사용하여 센서데이터 읽기

5-3. int MotionDetection()

역할 : 모션센서를 위한 센서데이터 읽기

5-4. int SensorSetting()

사용한 메소드 : 5-5 | 5-6

5-5. int SetPi()

-wiringPi 세팅

5-6. int SetSpi()

-wiringSPI 세팅

---------------------------------------------------------------------------

GET,SET FUNCTION(5-7 ~ 5-14)

5-7. void SetSensorType(int type)

5-8. int GetMcpData()

5-9. float GetVolt()

5-10. void SetVolt(float vol)

5-11. float GetRealData()

5-12. void SetRealData(float realValue)

5-13. bool GetTimerOn()

5-14. void SetTimerOn(bool on)

5-15. int GetElectronicOnTime()

5-16. clock\_t GetStartTime()

---------------------------------------------------------------------------

5-17. void RelaySetting()

-릴레이 제어를 위한 세팅

5-18. void ElectronicOn()

-사물기기 작동

5-19. void ElectronicOff()

-사물기기 정지

5-20. int ReadMcpData()

사용한 메소드 : 5-2

역할 : mcp-3208을 사용하는 센서데이터 디지털값 읽어오기

5-21. int ReadMotionData()

사용한 메소드 : 5-3

역할 : 모션 센서 데이터 읽어오기

5-22. float McpDataToVolt()

역할 : 디지털값을 볼트값으로 변환

5-23. virtual float VoltToRealData() = 0

역할 : 각 센서에 맞게 실용값으로 변환하기 위한 가상 함수(추상 메소드 개념)

5-24. void ShowData()

역할 : 읽은 데이터 출력

**========================================================**

**●SmartSensor.h**

#ifndef SMARTSENSOR\_H\_

#define SMARTSENSOR\_H\_

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

#include <string.h>

#include <sys/socket.h>

#include <sys/stat.h>

#include <arpa/inet.h>

#include <time.h>

#include "TCPClient.h"

#include "Sensor.h"

===========================================================================

**6. Smart Sensor Class -Client, Sensor 상속**

---------------------------------------------------------------------------

[필드 정보/ 이름 / 역할]

private:

char optimumMsg[BUFSIZ]

-홈매니저로부터 받은 적정값 메시지

float optimumData

-적정값 데이터를 실수로 변환한것

float tolerance

-적정값의 오차 범위(적정값 +- tolerance 가 적정 범위가 됨)

---------------------------------------------------------------------------

SMARTSENSOR MANAGE FUNCTION(1-1 ~ 1-32)

**[메소드 번호./ 메소드 이름/ 정보/해당 메소드 내에서 사용한 메소드/역할**

public:

6-1. SmartSensor()

6-2. ~SmartSensor()

6-3. void CommunicationMsgSetting(int type)

사용한 메소드 : 4-13

역할 : 다른 스마트센서 데이터를 받기위한 메시지 설정 ====> 해당 메소드 제거 예정 (주고받는 메시징 형식이 바뀌었음.. ip주소까지 추가 되었으므로 알고싶은 스마트 기기의 ip주소를 알아야 데이터를 받을수 있다..)

6-4. void OptimumMsgToData()

역할 : 받은 적정값 문자열을 실수형으로 바꿈

6-5. void Sleep()

역할 : 센싱간격을 늘이기위한 sleep()

6-6. void ReceiveOptimumMsg()

사용한 메소드 : 4-20 | 4-2 | 4-3 | 4-4 | 4-6 | 6-9 | 6-4

역할 : 홈매니저로부터 자신 센서 타입에 맞는 적정값 메시지 받음

---------------------------------------------------------------------------

GET,SET FUNCTION(6-7 ~ 6-12)

6-7. char\* GetOptimumMsg()

6-8. void SetOptimumMsg(char optimum[])

6-9. float GetOptimumData()

6-10. void SetOptimumData(float optData)

6-11. float GetTolerance()

6-12. void SetTolerance(float tol)

---------------------------------------------------------------------------

VIRTUAL FUNCTION(6-13 ~ 6-19)

6-13. virtual float VoltToRealData() = 0

역할 : 5-23과 같음

6-14. virtual void SendMsgSetting(float data)=0

역할 : float 형의 data를 문자열로 바꾸어 보낼 메시지 세팅

6-15. virtual void SendMyData()=0

역할 : 홈매니저에게 자신의 데이터 보냄

6-16. virtual void ReceiveOtherData(int type)=0

역할 : 다른 스마트센서의 데이터를 받음 ==> 없앨 예정..이유는 6-3과 같음

6-17. virtual float MyRealData()=0

역할 : 자신의 실용값 가져오기

6-18. virtual void Decision()=0

역할 : 센서데이터와 적정값을 비교하여 사물기기 제어

6-19. virtual void CalculateDecision() = 0

역할 : decision을 위한 적정값과 센서테이터 연산

아래 클래스는 각 센서 타입에 맞게 SmartSensor클래스의 추상메소드를 구현한 클래스들

===========================================================================

**7. Temp Smart Sensor Class -SmartSensor 상속**

---------------------------------------------------------------------------

public:

7-1. TempSmartSensor()

사용한 메소드 : 6-12 | 5-7 | 4-12

7-2.virtual float VoltToRealData()

사용한 메소드 : 5-8 | 5-12

역할 : 5-23과 같음

7-3. virtual void SendMsgSetting(float data)

사용한 메소드 : 4-21

역할 : 6-14 와 같음

7-4. virtual void SendMyData()

사용한 메소드 : 4-20 | 7-3

역할 : 6-15 와 같음

7-5. virtual void ReceiveOtherData(int type)

사용한 메소드 : 6-3 | 4-20 | 7-3

역할 : 6-16 와 같음

7-6. virtual float MyRealData()

사용한 메소드 : 5-20 | 7-2 | 5-11

역할 : 6-17 와 같음

7-7. virtual void Decision()

사용한 메소드 : 6-9 | 7-8

역할 : 6-18 와 같음

7-8. virtual void CalculateDecision()

사용한 메소드 : 5-11 | 6-9 | 6-11 | 5-18 | 5-19 | 5-13 | 5-14

역할 : 6-19 와 같음

===========================================================================

**8. Humd Smart Sensor Class -SmartSensor 상속**

--------------------------------------------------------------------------

public:

8-1. HumdSmartSensor()

사용한 메소드 : 6-12 | 5-7 | 4-12

8-2. virtual float VoltToRealData()

사용한 메소드 : 5-8 | 5-12

역할 : 5-23과 같음

8-3. virtual void SendMsgSetting(float data)

사용한 메소드 : 4-21

역할 : 6-14 와 같음

8-4. virtual void SendMyData()

사용한 메소드 : 4-20 | 8-3

역할 : 6-15 와 같음

8-5. virtual void ReceiveOtherData(int type)

사용한 메소드 : 6-3 | 4-20 | 8-3

역할 : 6-16 와 같음

8-6. virtual float MyRealData()

사용한 메소드 : 5-20 | 8-2 | 5-11

역할 : 6-17 와 같음

8-7. virtual void Decision()

사용한 메소드 : 6-9 | 8-8

역할 : 6-18와 같음

8-8. virtual void CalculateDecision()

사용한 메소드 : 5-11 | 6-9 | 6-11 | 5-18 | 5-19 | 5-13 | 5-14

역할 : 6-19 와 같음

===========================================================================

**9. Lux Smart Sensor Class -SmartSensor 상속**

---------------------------------------------------------------------------

public:

9-1. LuxSmartSensor()

사용한 메소드 : 6-12 | 5-7 | 4-12

9-2. virtual float VoltToRealData()

사용한 메소드 : 5-8 | 5-12

역할 : 5-23과 같음

9-3. virtual void SendMsgSetting(float data)

사용한 메소드 : 4-21

역할 : 6-14 와 같음

9-4. virtual void SendMyData()

사용한 메소드 : 4-20 | 9-3

역할 : 6-15 와 같음

9-5. virtual void ReceiveOtherData(int type)

사용한 메소드 : 6-3 | 4-20 | 9-3

역할 : 6-16 와 같음

9-6. virtual float MyRealData()

사용한 메소드 : 5-20 | 9-2 | 5-11

역할 : 6-17 와 같음

9-7. virtual void Decision()

사용한 메소드 : 6-9 | 9-8

역할 : 6-18 와 같음

9-8. virtual void CalculateDecision()

사용한 메소드 : 5-11 | 6-9 | 6-11 | 5-18 | 5-19 | 5-13 | 5-14

역할 : 6-19 와 같음

===========================================================================

**10. Motion Smart Sensor Class -SmartSensor 상속**

---------------------------------------------------------------------------

public:

10-1. MotionSmartSensor()

사용한 메소드 : 6-12 | 5-7 | 4-12

10-2. virtual float VoltToRealData()

사용한 메소드 : 5-8 | 5-12

역할 : 5-23과 같음

10-3. virtual void SendMsgSetting(float data)

사용한 메소드 : 4-21

역할 : 6-14 와 같음

10-4. virtual void SendMyData()

사용한 메소드 : 4-20 | 10-3

역할 : 6-15 와 같음

10-5. virtual void ReceiveOtherData(int type)

사용한 메소드 : 6-3 | 4-20 | 10-3

역할 : 6-16 와 같음

10-6. virtual float MyRealData()

사용한 메소드 : 5-20 | 10-2 | 5-11

역할 : 6-17 와 같음

10-7. virtual void Decision()

사용한 메소드 : 6-9 | 10-8

역할 : 6-18 와 같음

10-8. virtual void CalculateDecision()

사용한 메소드 : 5-11 | 6-9 | 6-11 | 5-18 | 5-19 | 5-13 | 5-14

역할 : 6-19 와 같음