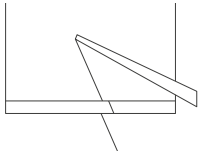
Bring ideas to life
VIA University College



From Design to C Part I

ESW1

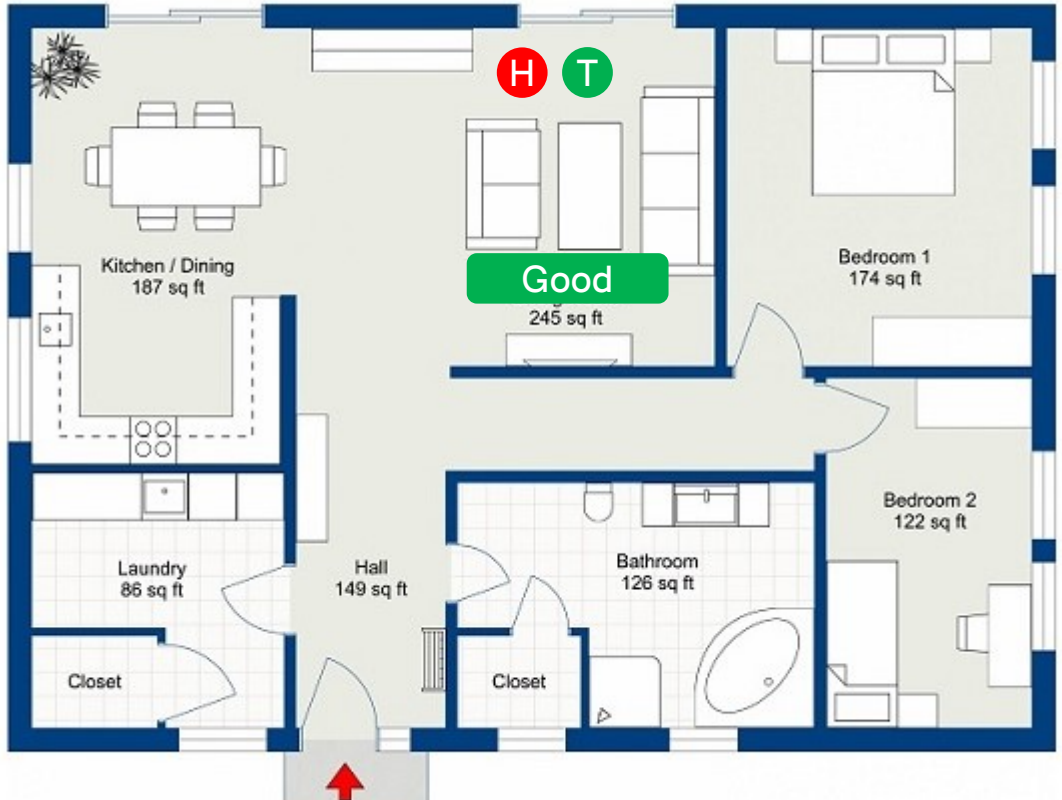
From Design to C

How to come from a Design (UML) to C-code

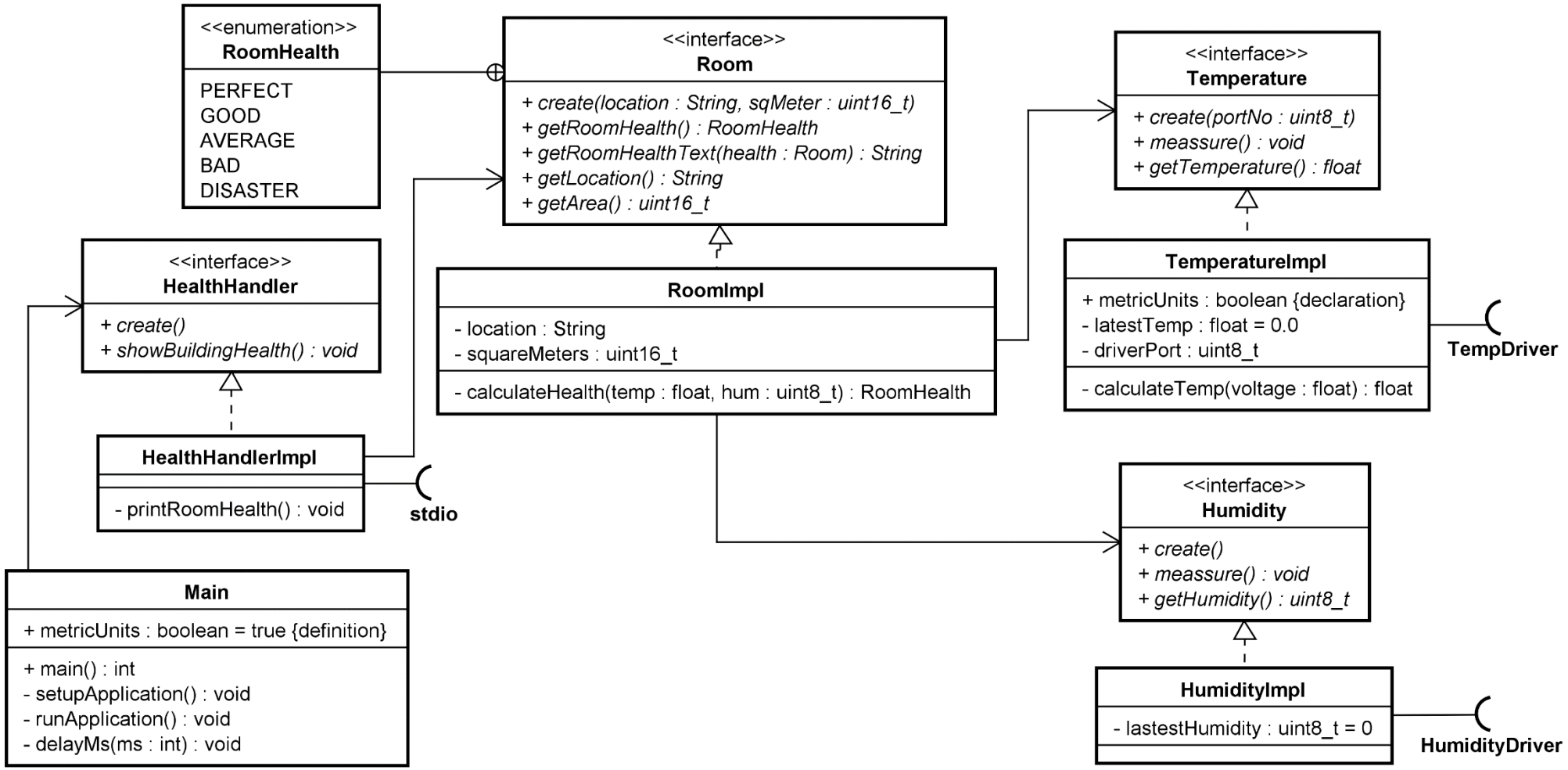
The following example design is for a small application that measures and shows the environment (health) in a single room (living room) in a building

The health is based on a temperature- and a humidity-measuring in the room.
The health should be shown once per second

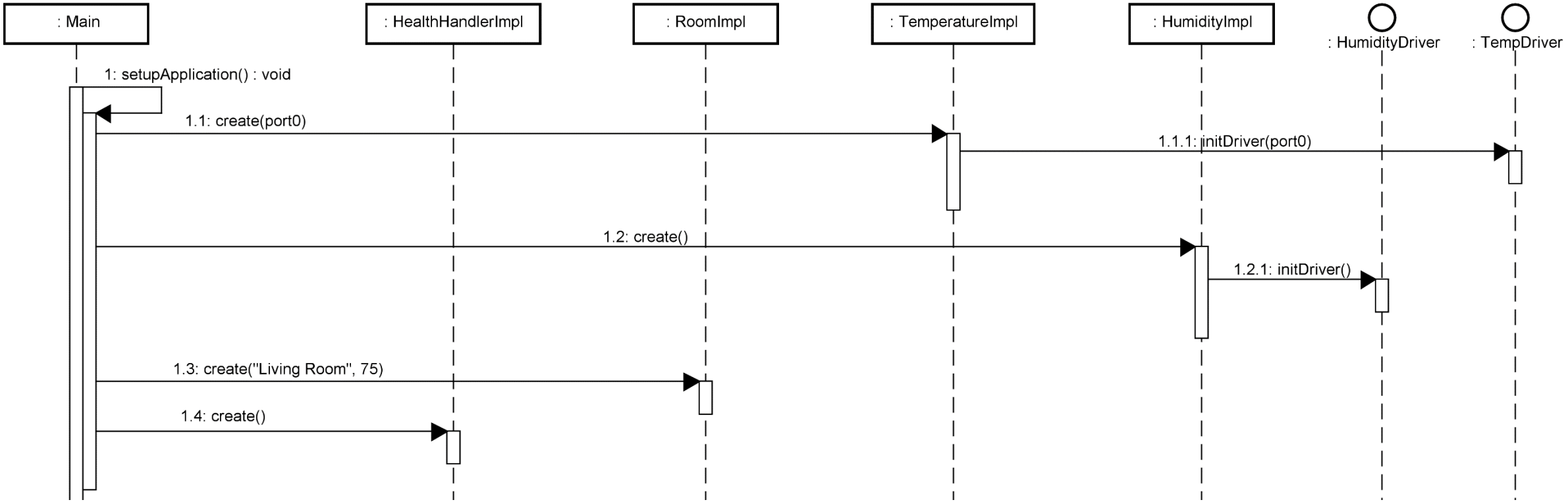
The sensors are all simulated in this example



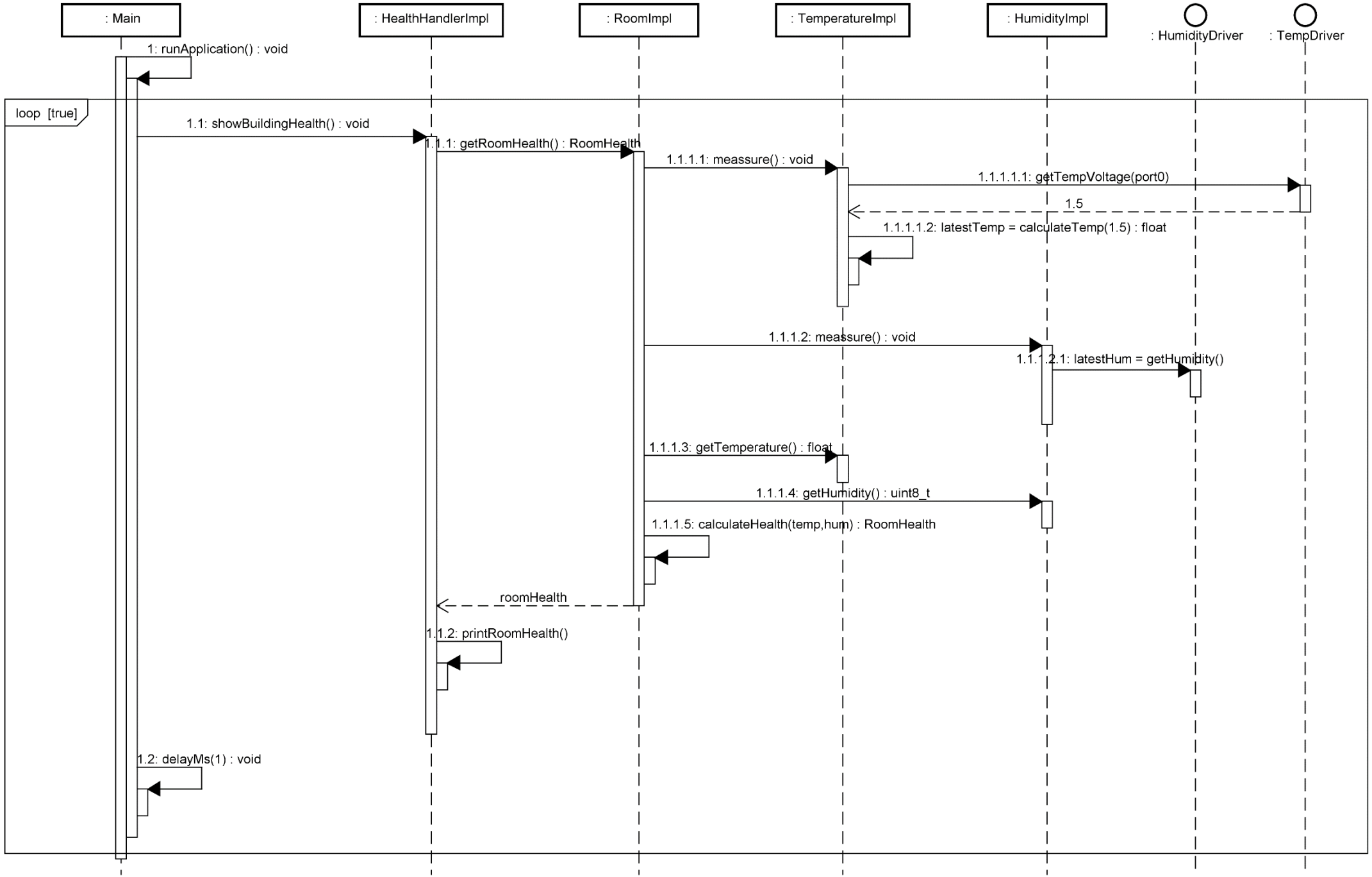
An Example Without Classes



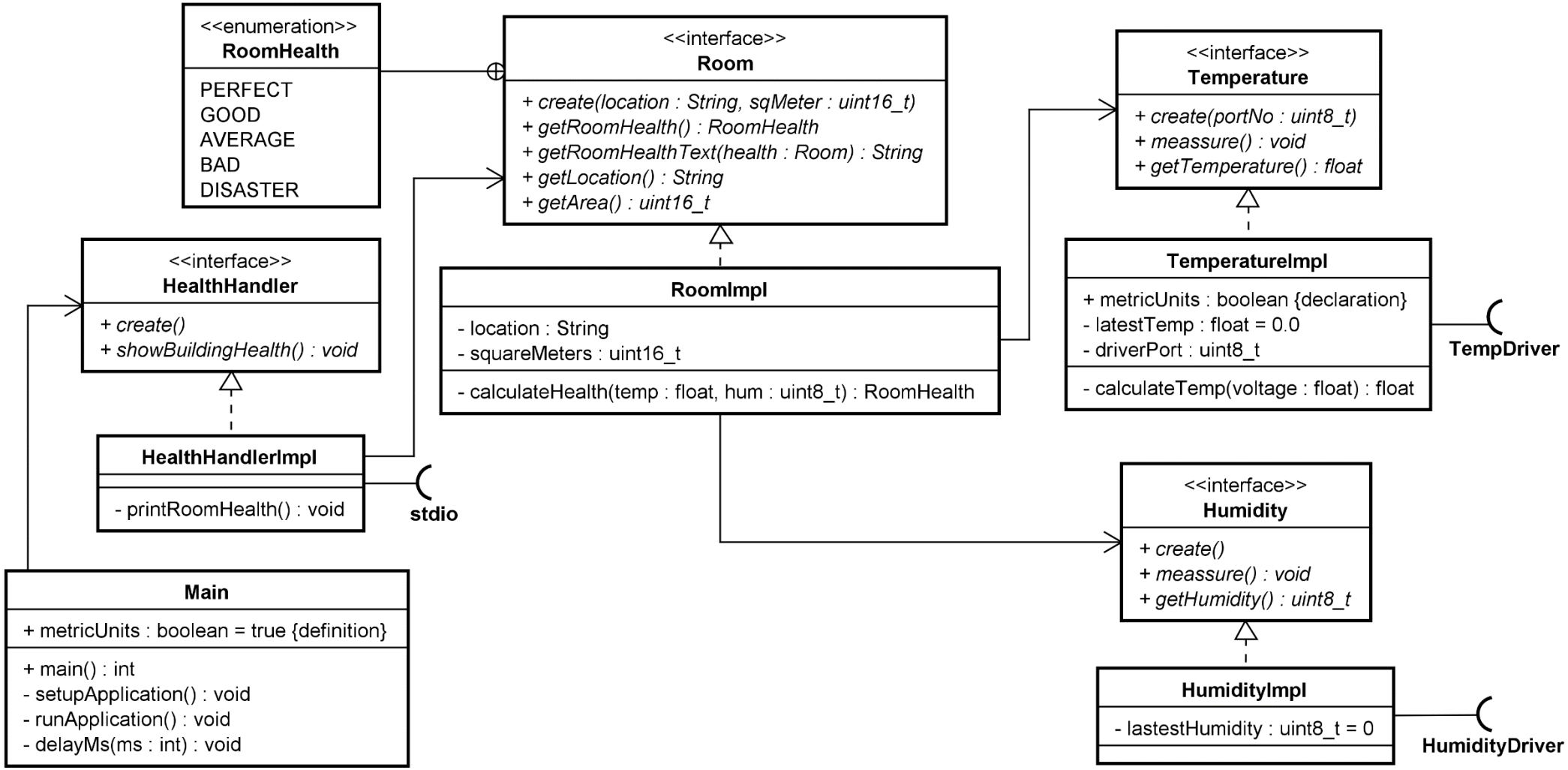
An Example
Without Classes – Setup Application scenario



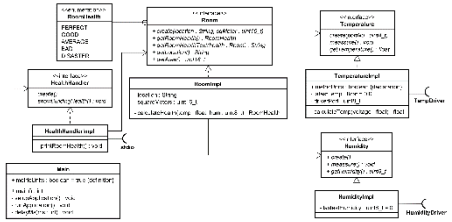
An Example - Without Classes
Run Application scenario

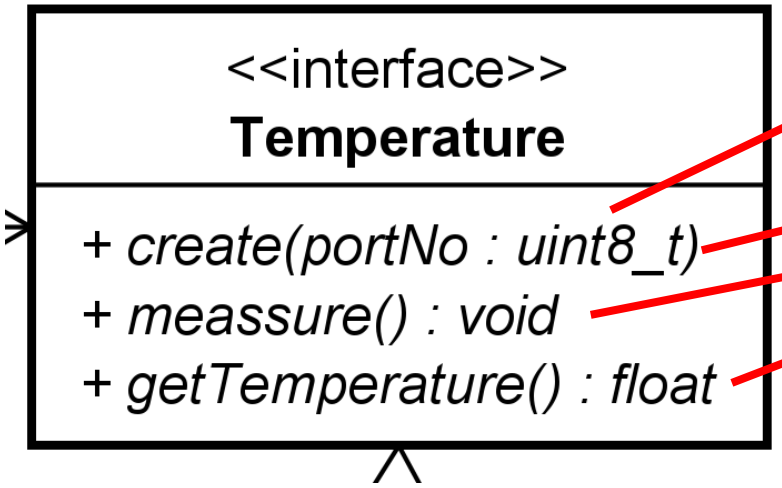


An Example Without Classes



An Example Without Classes





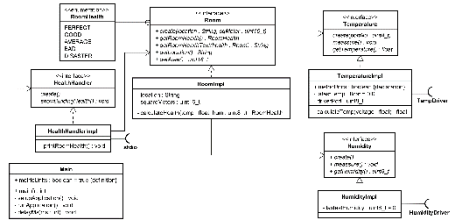
temperature.h

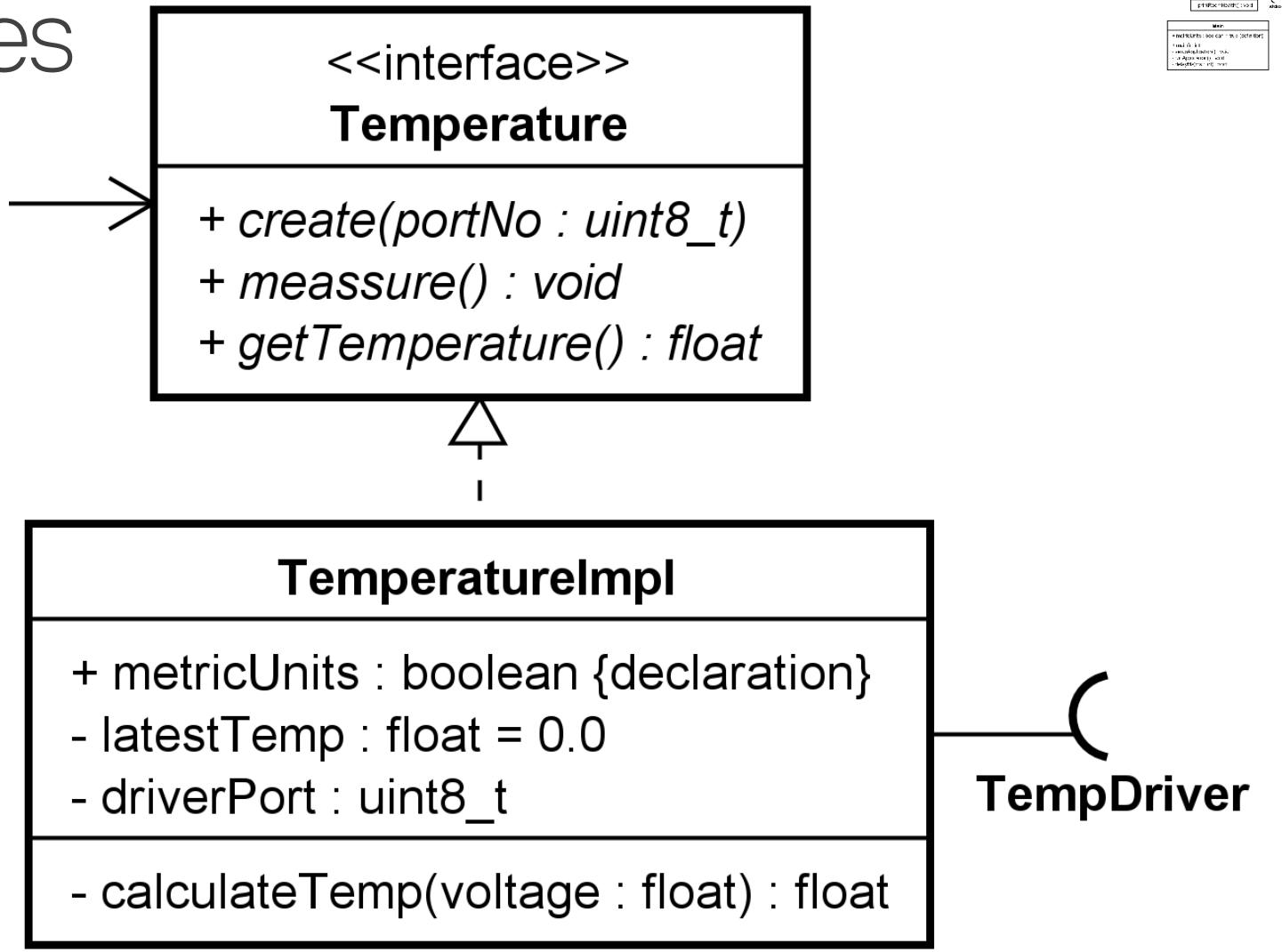
```
#pragma once
#include <stdint.h>

void temperature_create(uint8_t portNo);
void temperature_meassure(void);
float temperature_getTemperature(void);
```

Why are the prefix **`temperature_`** used?

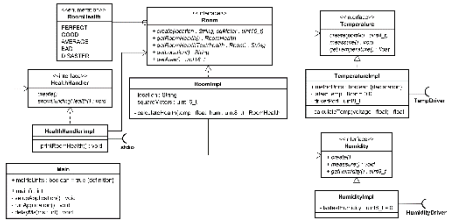
- A kind of namespace to make the names unique in the application
  - Use the name of the interface as the prefix

An Example Without Classes

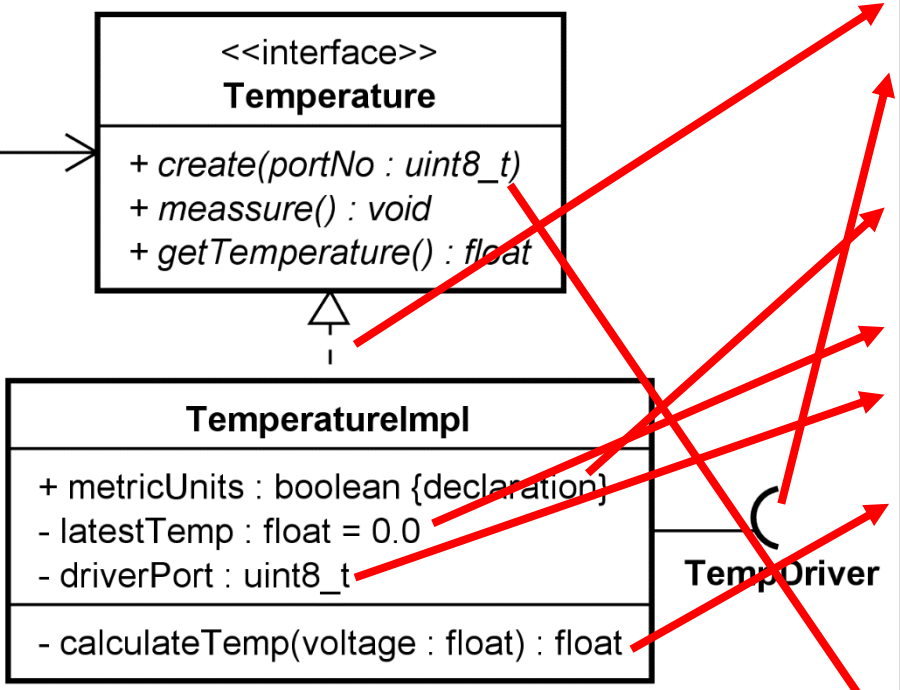




An Example Without Classes



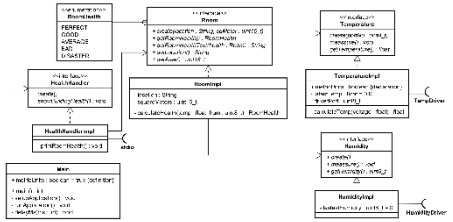
temperature.c



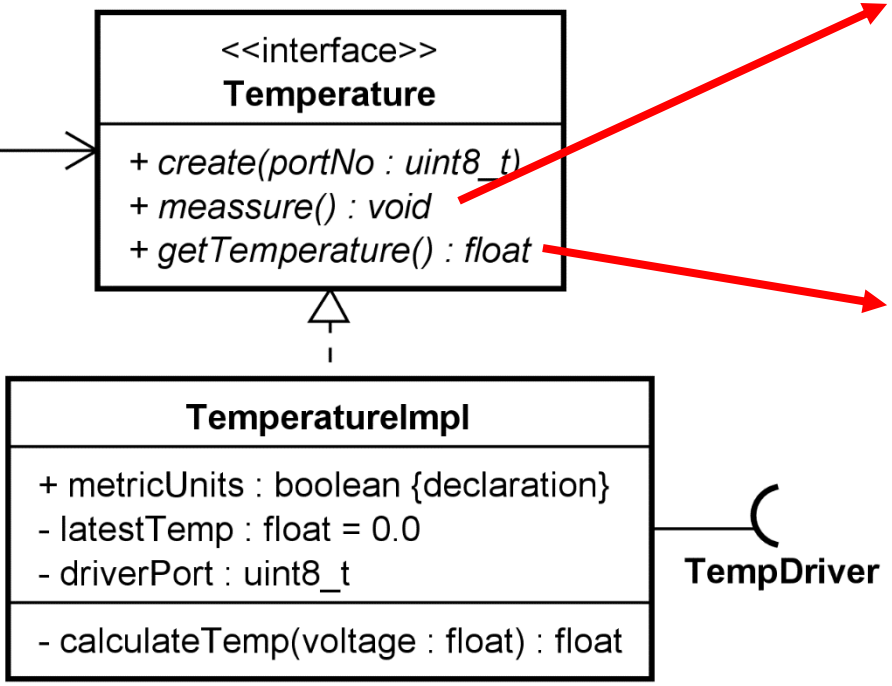

```c
#include "temperature.h"
#include <tempDriver.h>

extern bool temperature_metricUnits; // Declaration

static float _latestTemp = 0.0;
static uint8_t _driverPort;

static float _calculateTemp(float voltage) {
   return 15.0+(voltage * 5.0); // dummy calc
}

void temperature_create(uint8_t portNo) {
    _driverPort = portNo;
    temperatureDriver_initDriver(portNo);
}
```

`static` means module (file) scope
Kind of `private`

An Example Without Classes

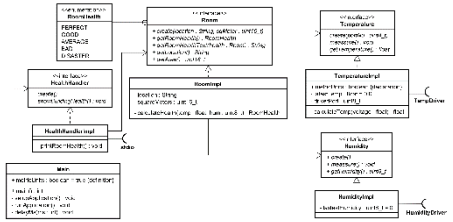


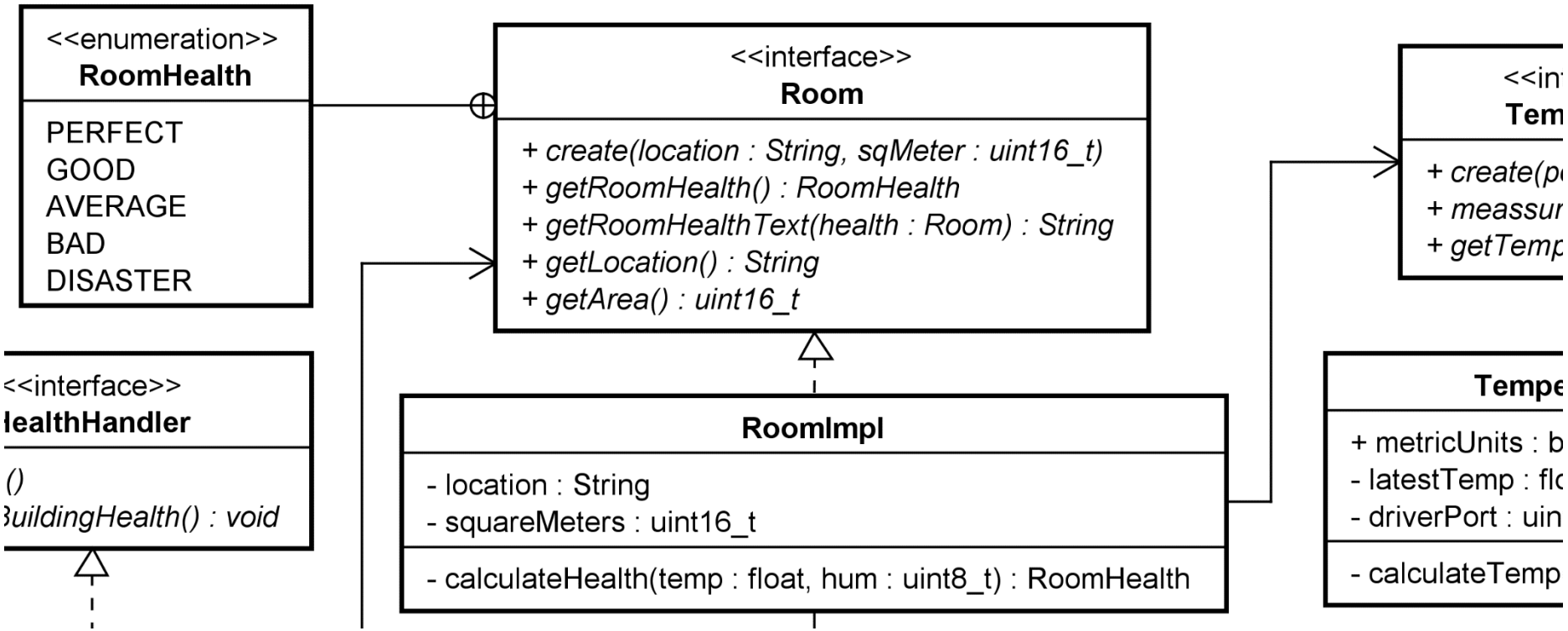
temperature.c - continued



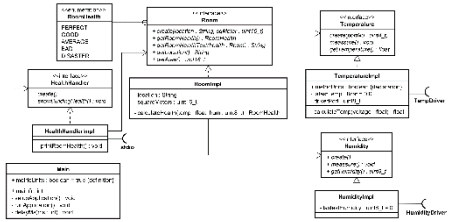

```c
void temperature_meassure(void) {
    _latestTemp =
    _calculateTemp(temperatureDriver_getVoltage());
}

float temperature_getTemperature(void){
    float _tmp = _latestTemp;
    if (temperature_metricUnits) {
        _tmp *= 0.2; // dummy conversion
    }

    return _tmp;
}
```

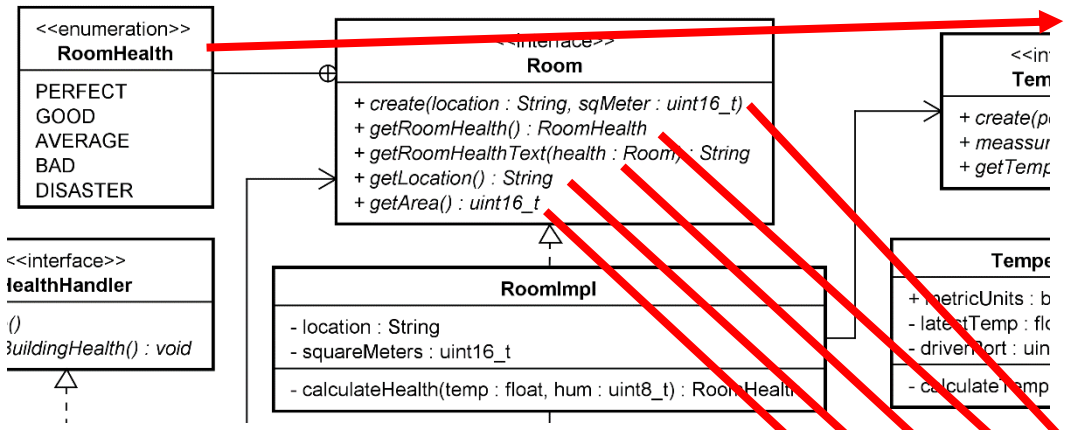
An Example Without Classes





An Example Without Classes

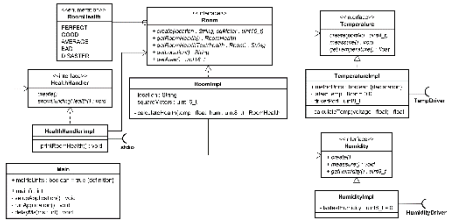




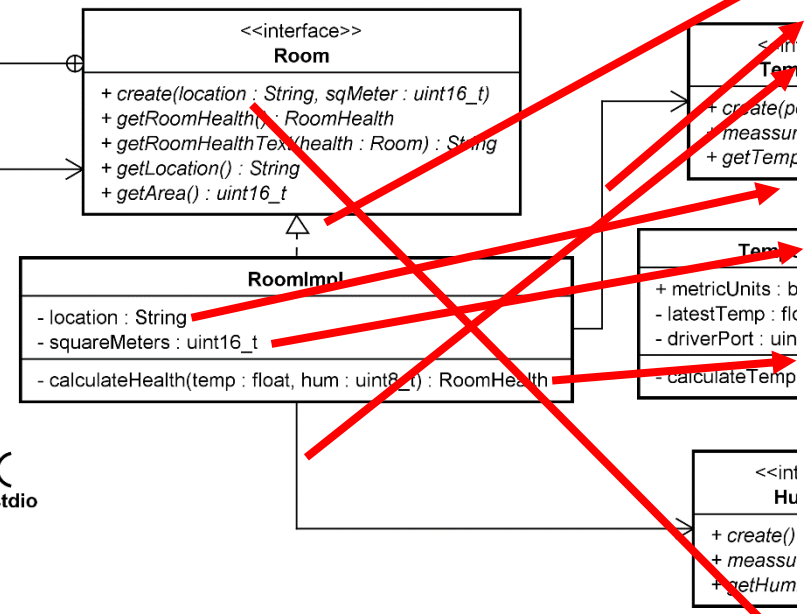
room.h

```c
#pragma once
#include <stdint.h>
typedef enum {
    PERFECT
    ,GOOD
    ,AVERAGE
    ,BAD
    ,DISASTER
}room_roomHealth_t;

void room_create(char* location, uint16_t sqMeter);
room_roomHealth_t room_getRoomHealth(void);
char* room_getRoomHealthText(room_roomHealth_t health);
char* room_getLocation(void);
uint16_t room_getArea(void);
```

An Example Without Classes

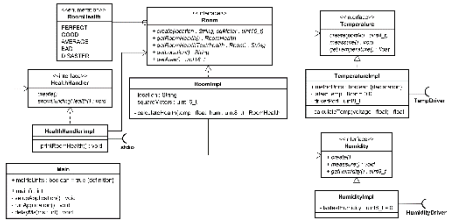


room.c

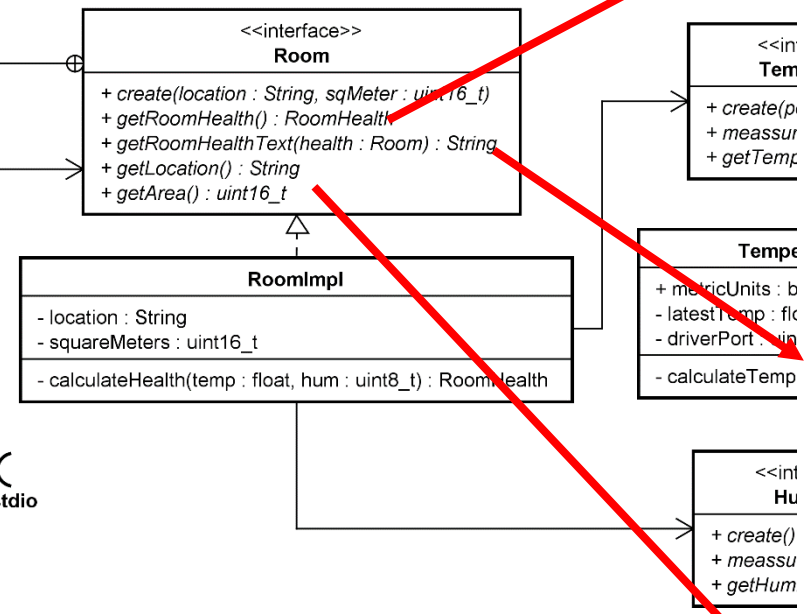



```c
#include <string.h>
#include "room.h"
#include "temperature.h"
#include "humidity.h"

static char _location[30] = { 0 };
static uint16_t _squareMeters;

static room_roomHealth_t _calculateHealth(float temp, uint8_t hum)
{
    return (uint16_t)(temp * hum) % 5;// dummy calculation;
}

void room_create(char* location, uint16_t sqMeter) {
   // Why does it look like this??
    strncpy(_location, location, sizeof(_location) - 1);
    _squareMeters = sqMeter;
}
```

An Example Without Classes

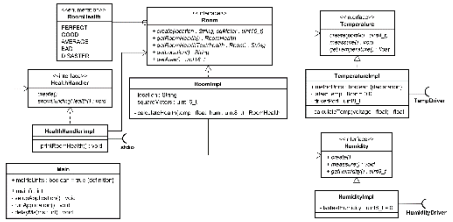


room.c - continued



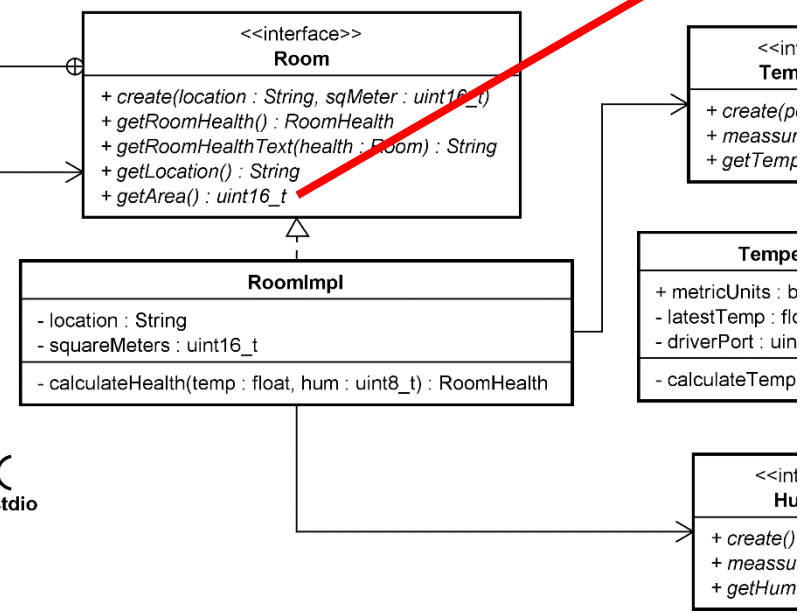

```c
room_roomHealth_t room_getRoomHealth(void) {
    temperature_meassure();
    humidity_meassure();

    return _calculateHealth(temperature_getTemperature(),
    humidity_getHumidity());
}

char* room_getRoomHealthText(room_roomHealth_t health) {
    static const char* _health_text[] = { "PERFECT", "GOOD",
    "AVERAGE", "BAD", "DISASTER" };
    return _health_text[health];
}

char* room_getLocation(void) {
    return _location;
}
```

An Example Without Classes

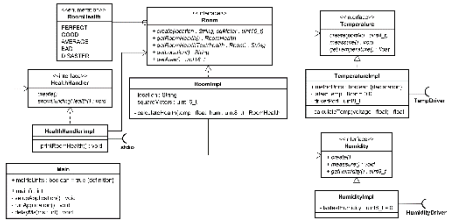


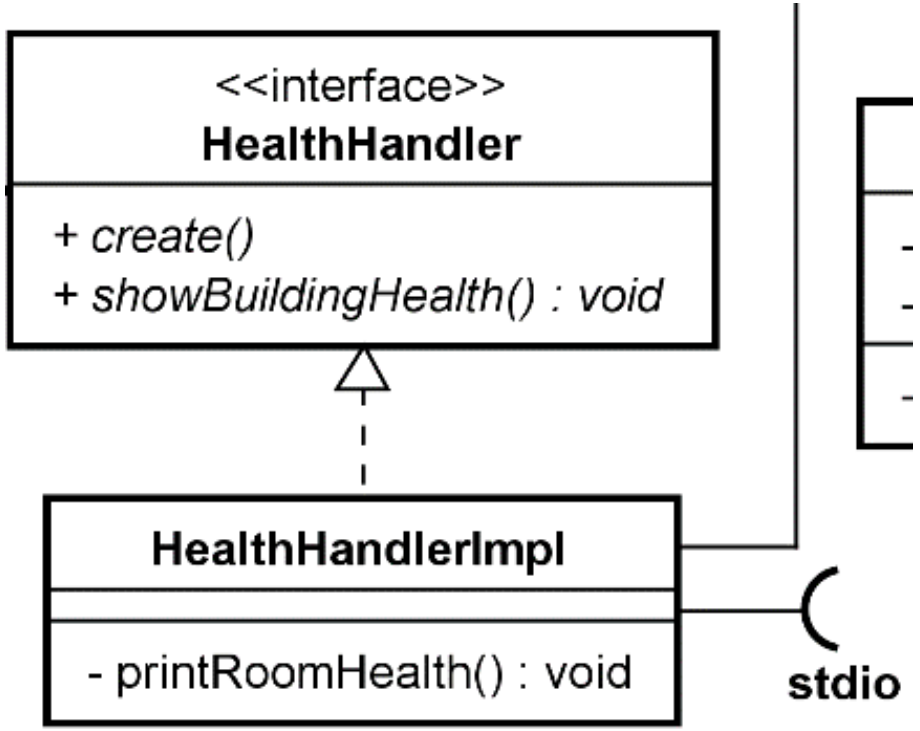
room.c - continued

```
uint16_t room_getArea(void) {
    return _squareMeters;
}
```

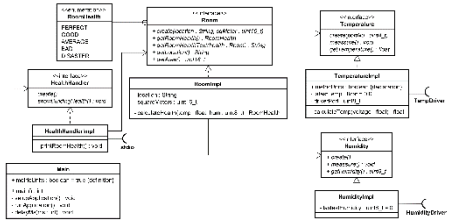


An Example Without Classes



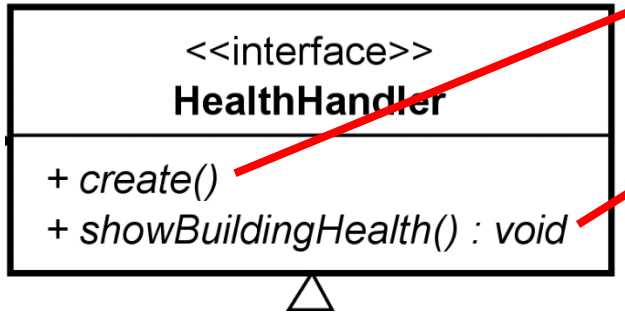


An Example Without Classes

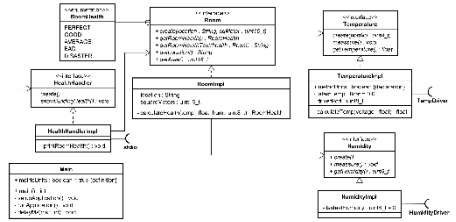


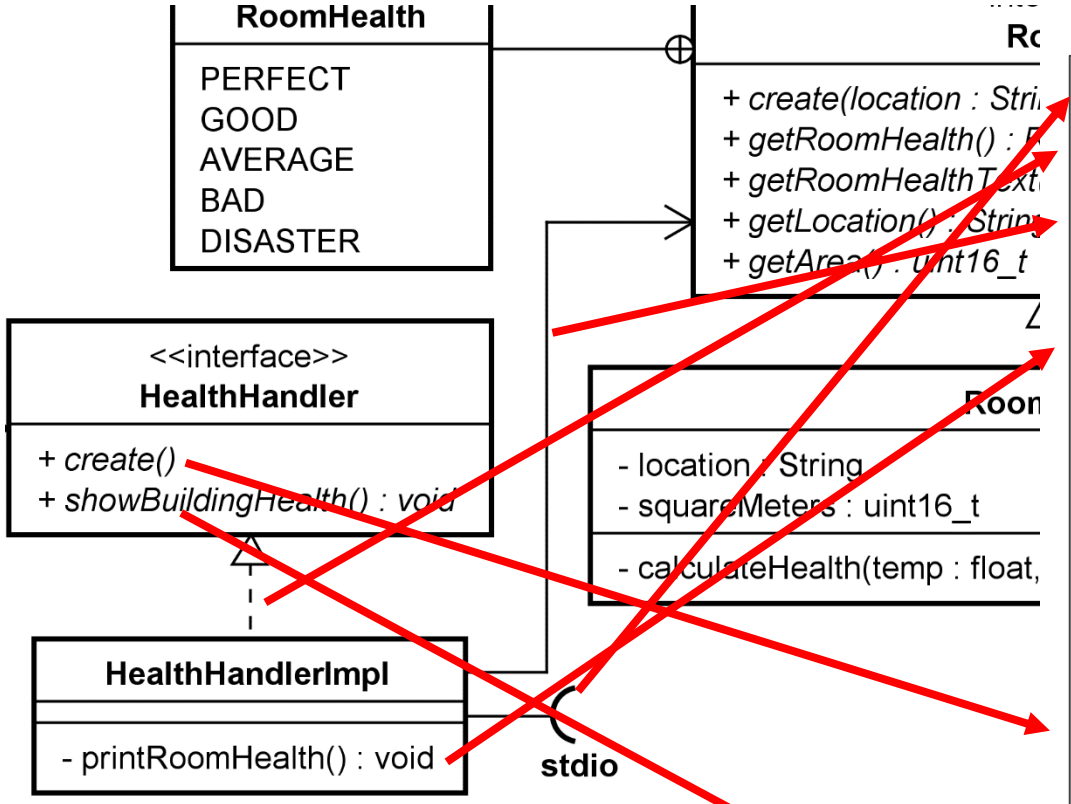
healthHandler.h

```
#pragma once
void healthHandler_create(void);
void healthHandler_showBuildingHealth(void);
```



An Example Without Classes

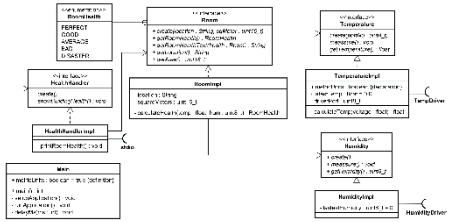




healthHandler.c

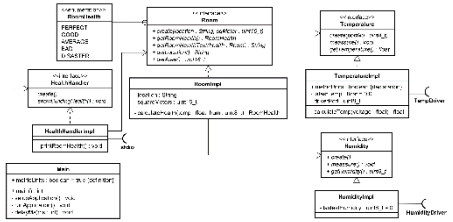
```c
#include <stdio.h>
#include "healtHandler.h"
#include "room.h"

static void _printRoomHealth(void) {
    printf("%s: area: %d m2 Health: %s\n",
    room_getLocation(), room_getArea(),
    room_getRoomHealthText(room_getRoomHealth()));
}

void healthHandler_create(void) {
    // do something
}

void healthHandler_showBuildingHealth(void) {
    _printRoomHealth();
}
```

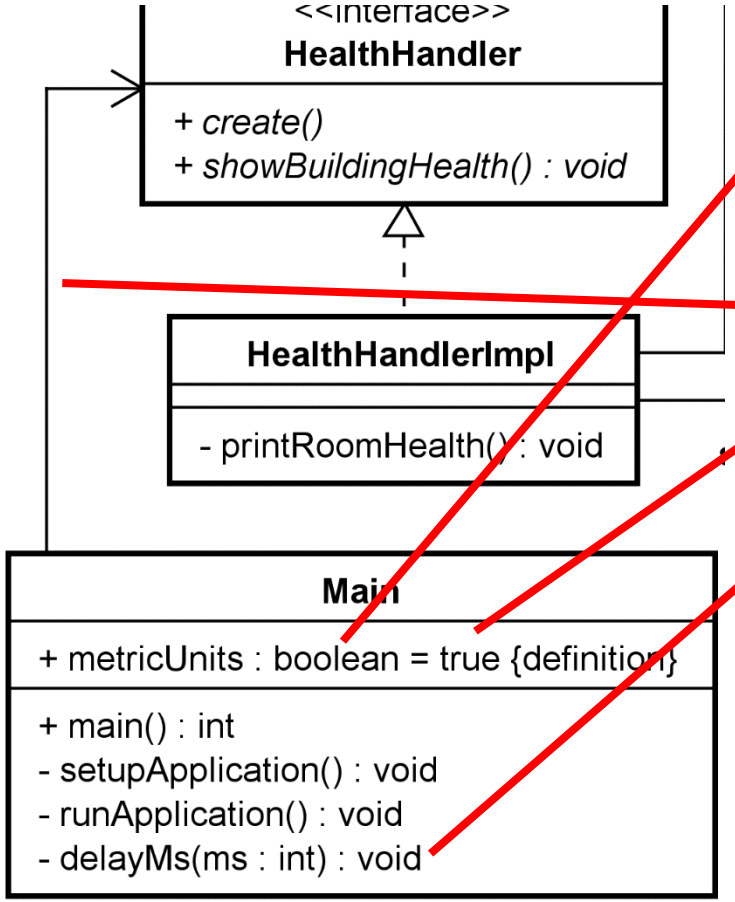
An Example Without Classes



Main
+ metricUnits : boolean = true {definition}
+ main() : int - setupApplication() : void - runApplication() : void - delayMs(ms : int) : void

An Example Without Classes



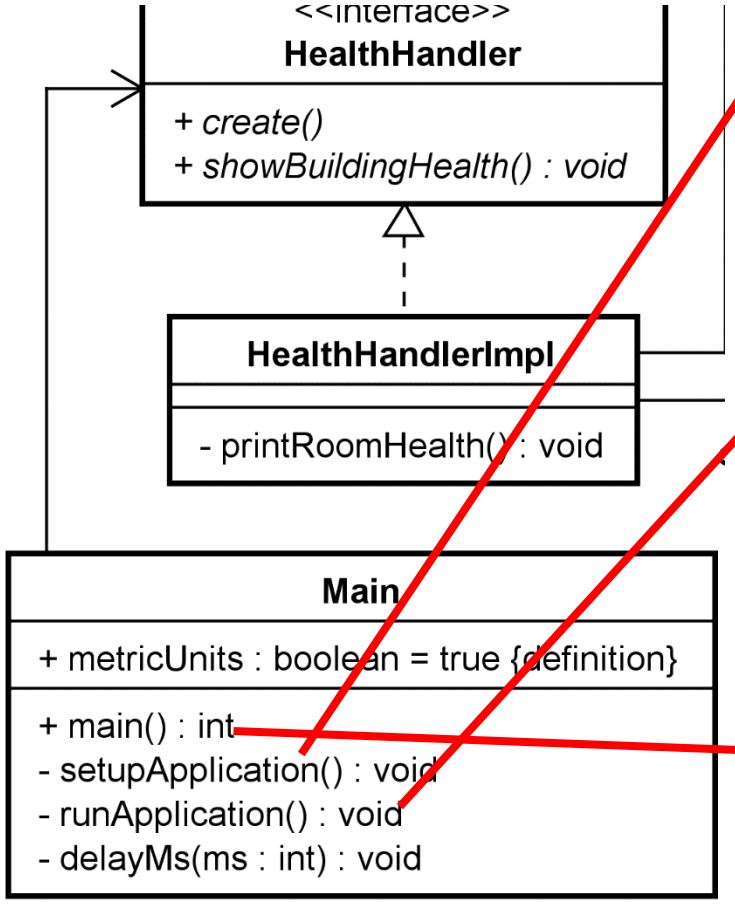


main.c

```
#include <time.h>
#include <stdbool.h>
#include "temperature.h"
#include "humidity.h"
#include "room.h"
#include "healtHandler.h"
bool temperature_metricUnits = true; // Definition

static void _delayMs(int milliseconds)
{
    long pause;
    clock_t now, then;

    pause = milliseconds * (CLOCKS_PER_SEC / 1000);
    now = then = clock();
    while ((now - then) < pause)
      now = clock();
}
```

To be able to initialise these modules

An Example Without Classes



main.c - continued

```
static void _setupApplication(void) {
    temperature_create(0);
    humidity_create();
    room_create("Living Room", 75);
    healthHandler_create();
}

static void _runApplication(void) {
    while (1) {
        healthHandler_showBuildingHealth();
        _delayMs(1000);
    }
}

int main(void) {
    _setupApplication();
    _runApplication();

    return 0;
}
```

Exercise – Going from Design to C-code

1. Design a **small** system using UML (**Class-** and **Sequence-Diagrams**)

   The criteria's for the design:
   - The system must be small – like the example I have shown today
   - The system does not need to give completely meaning
   - The system **must only use a multiplicity of one** between the classes

2. Check that your design can handle the wanted functionality

   - Check it by using your Sequence Diagrams

3. Implement the design in C