

OOAD:

#4 OOI

202211318 엄정석 / 202211334 이동훈 / 202211384 차민우 / 202211392 최환

Contents

- 1. 수정 사항**
 - a. OOA & OOD
- 2. 구현 (Coding)**
 - a. Pair Programming
 - b. Unit Test
 - c. Code Review
- 3. System Test**
 - a. Simulator
 - b. Test 수행 과정 및 결과
- 4. Static Code Analysis**
 - a. 정적분석 수행
 - b. 수행 결과

수정 사항

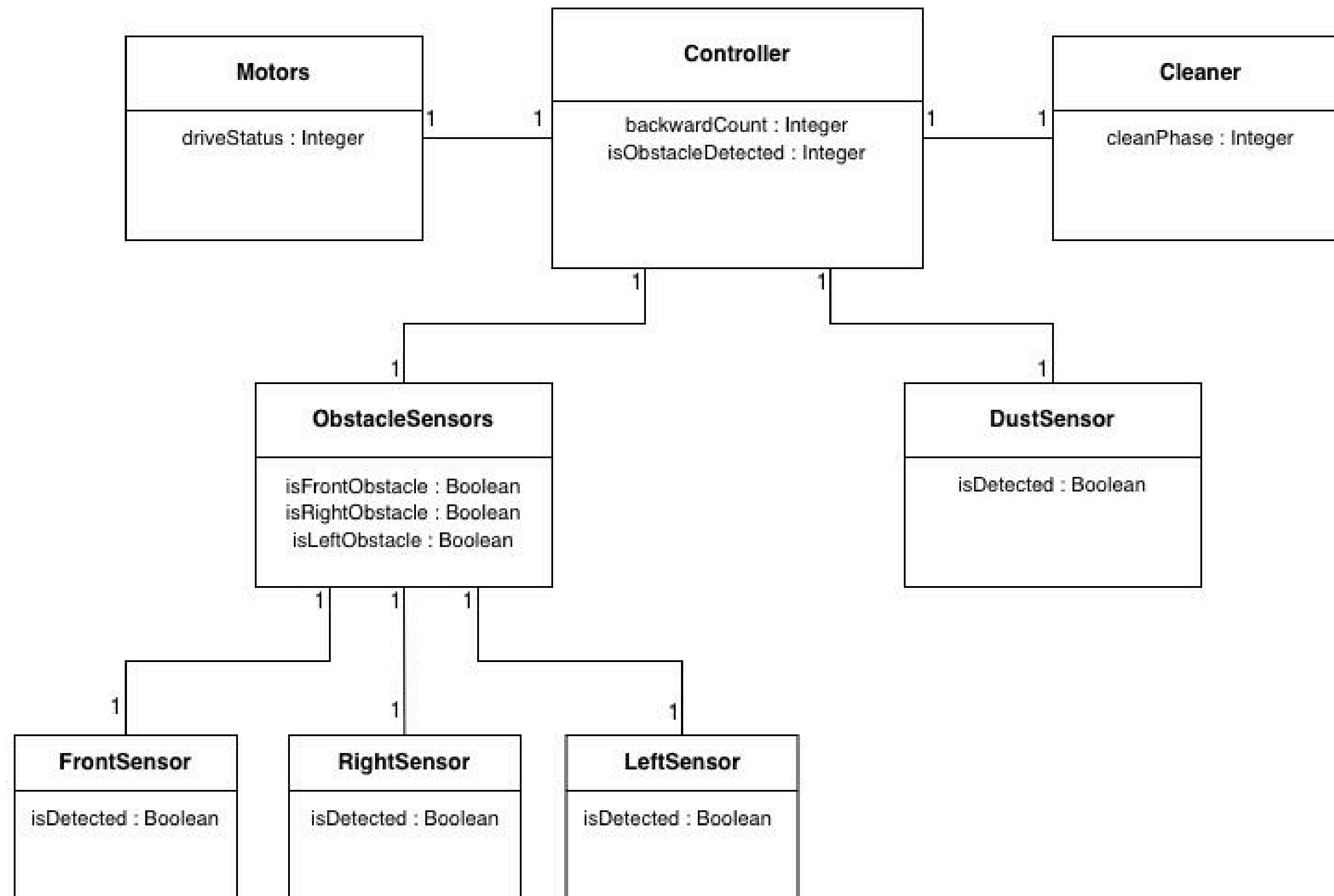
OOA & OOD

[OOA]
System Operations
(System Interface)

{Interface}
RVC SW

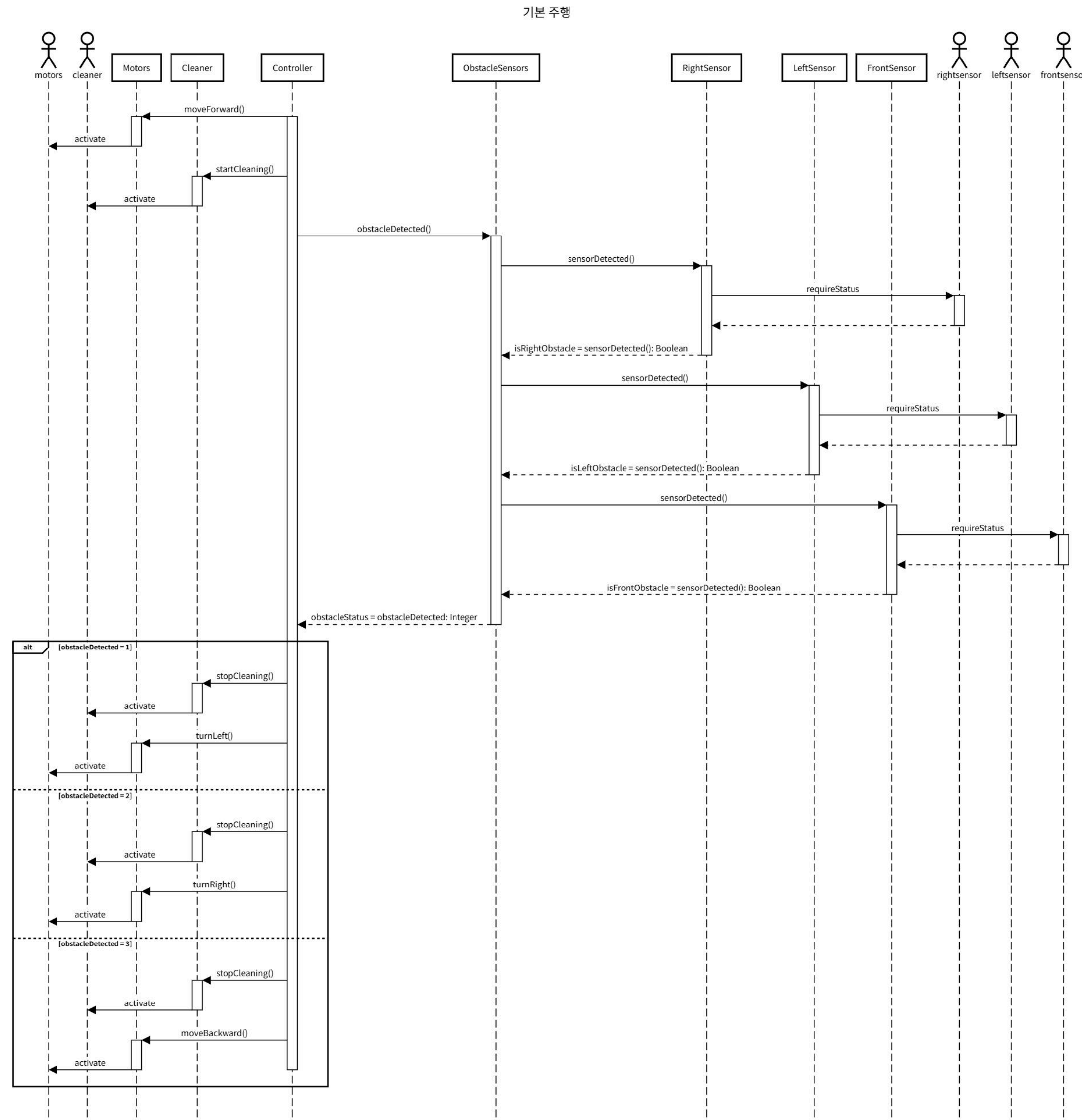
+powerOn()
+powerOff()

[OOA] Domain Model



[OOD] Sequence Diagrams

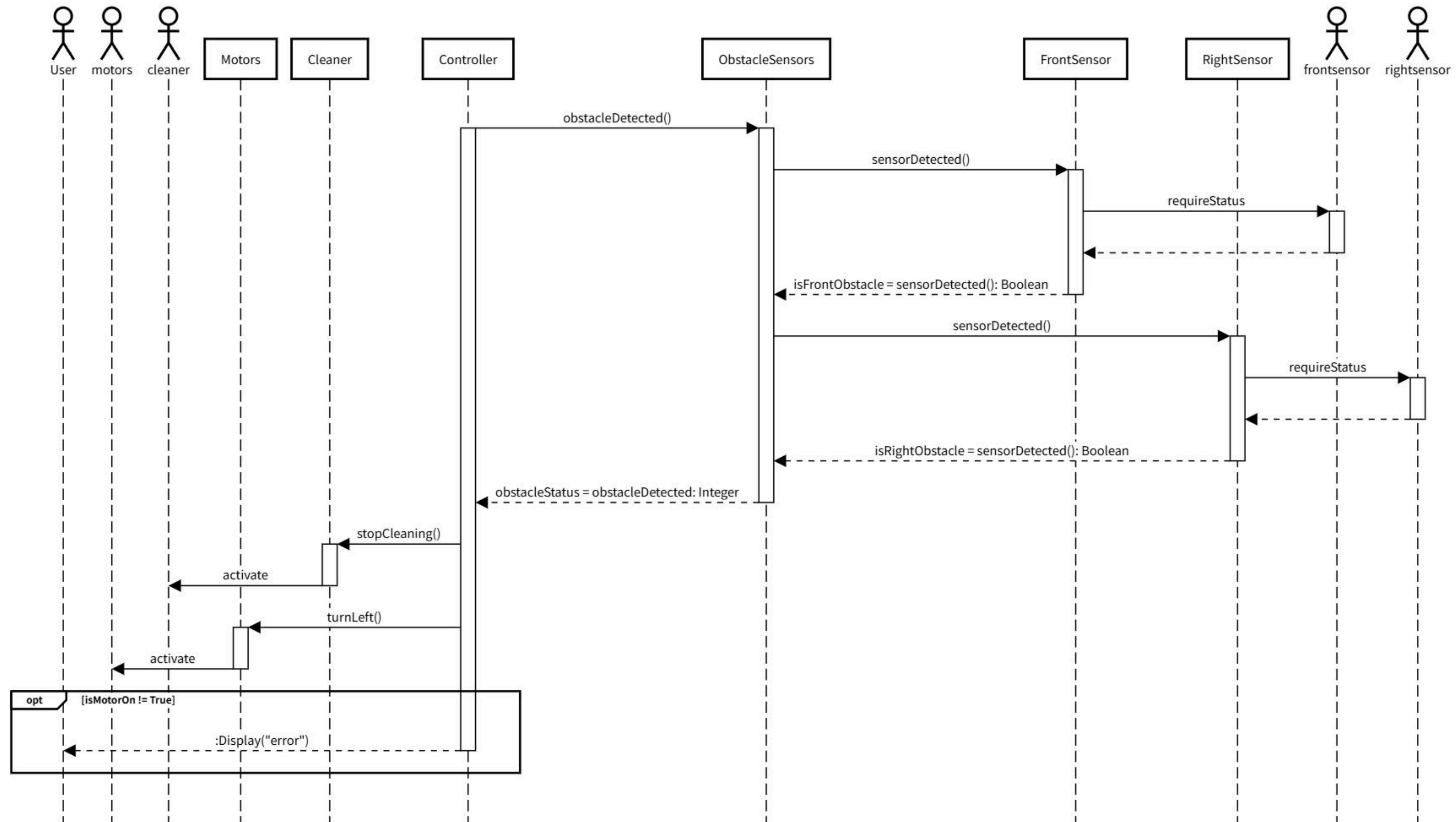
UC-03



[OOD] Sequence Diagrams

UC-04

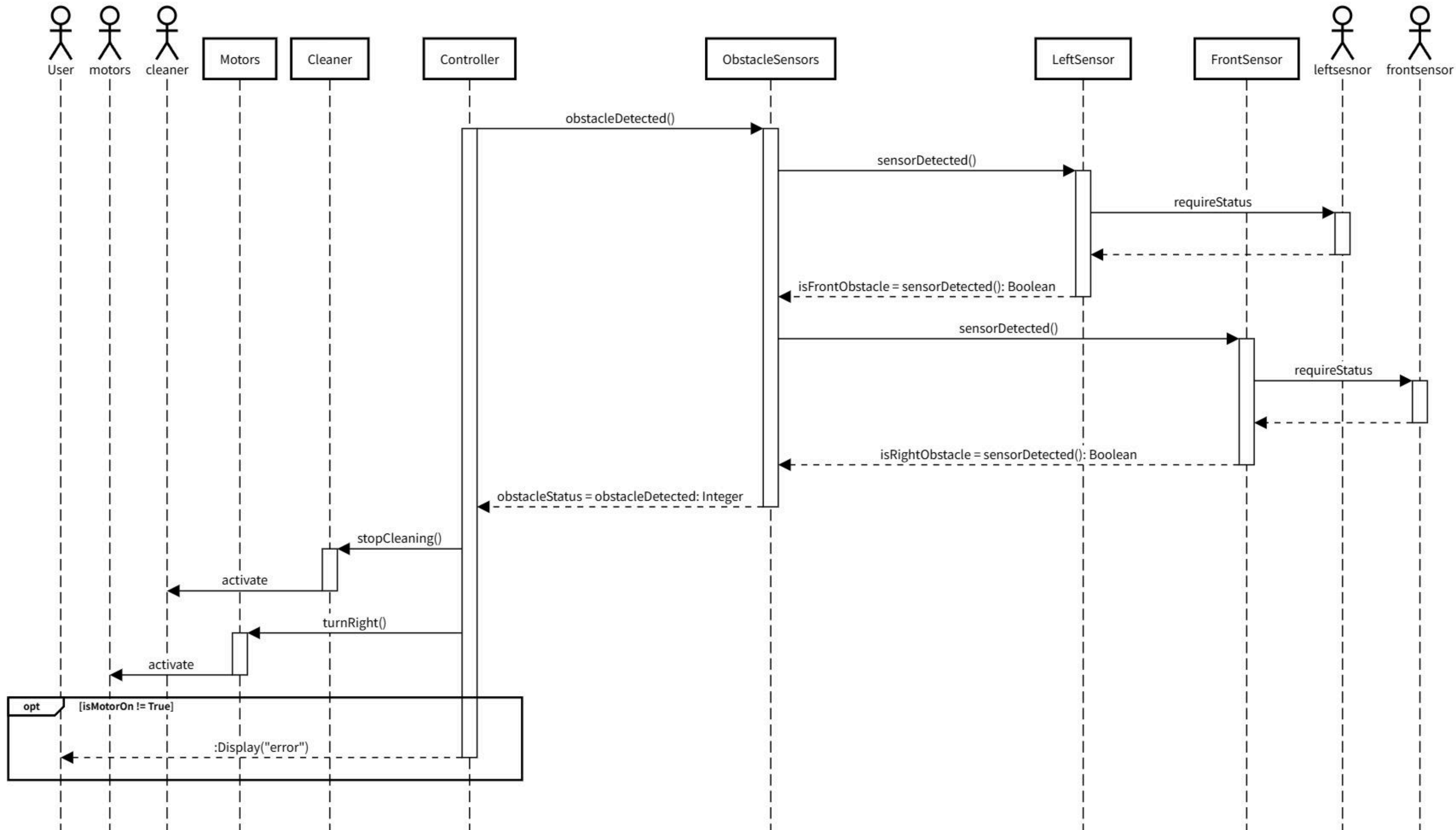
장애물 감지 - 좌회전



[OOD] Sequence Diagrams

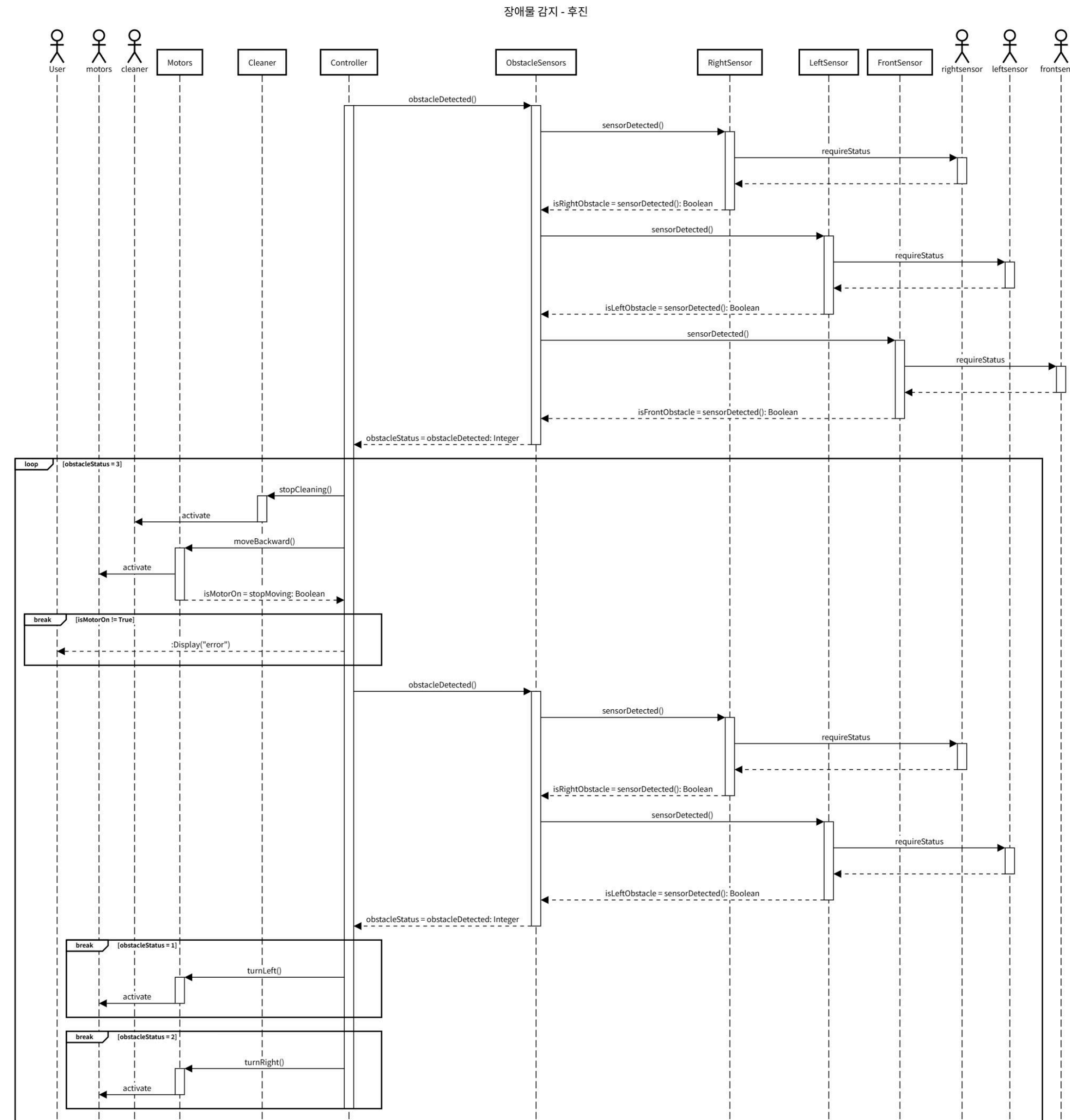
UC-05

장애물 감지 - 우회전



[OOD] Sequence Diagrams

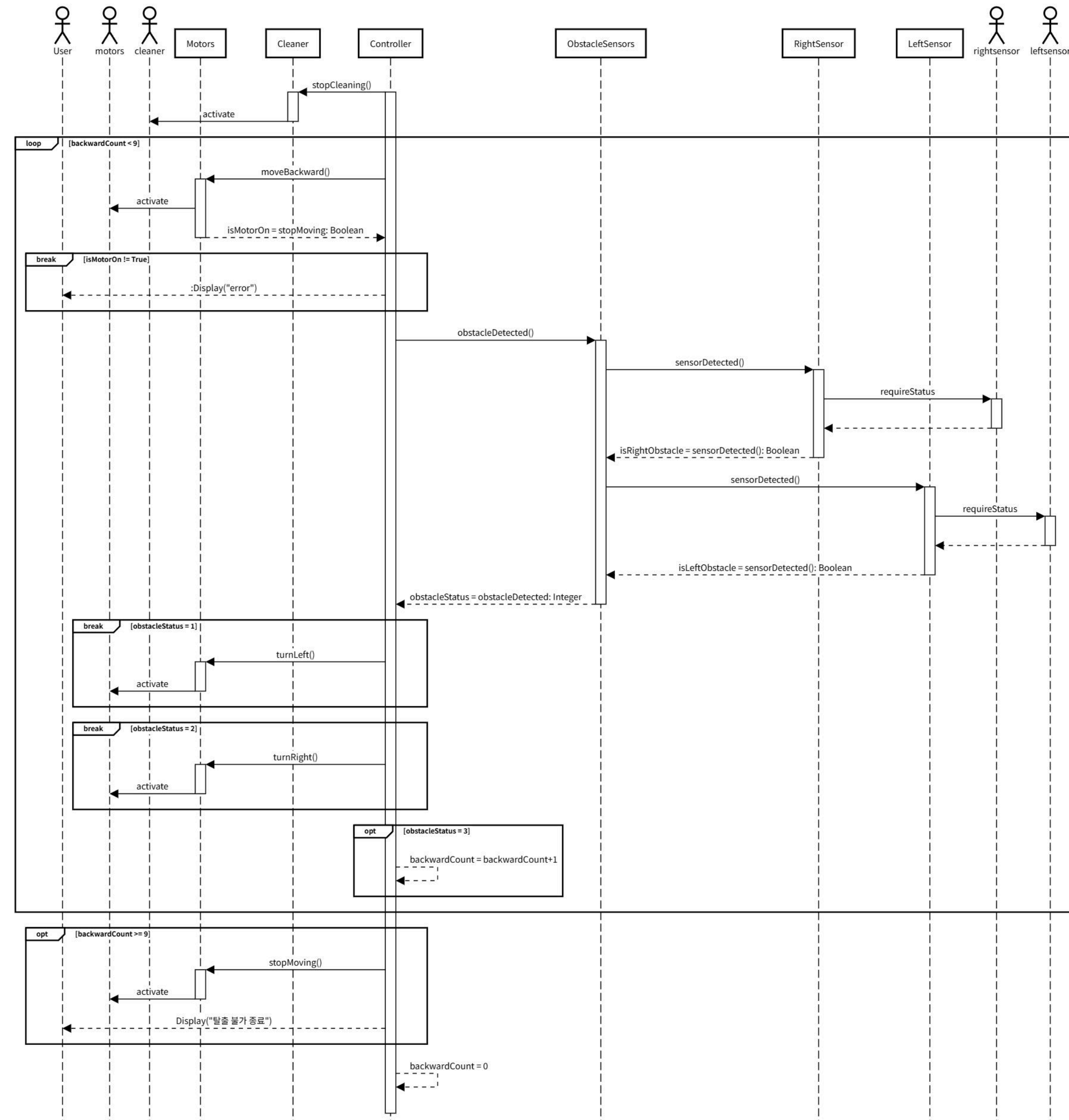
UC-06



[OOD] Sequence Diagrams

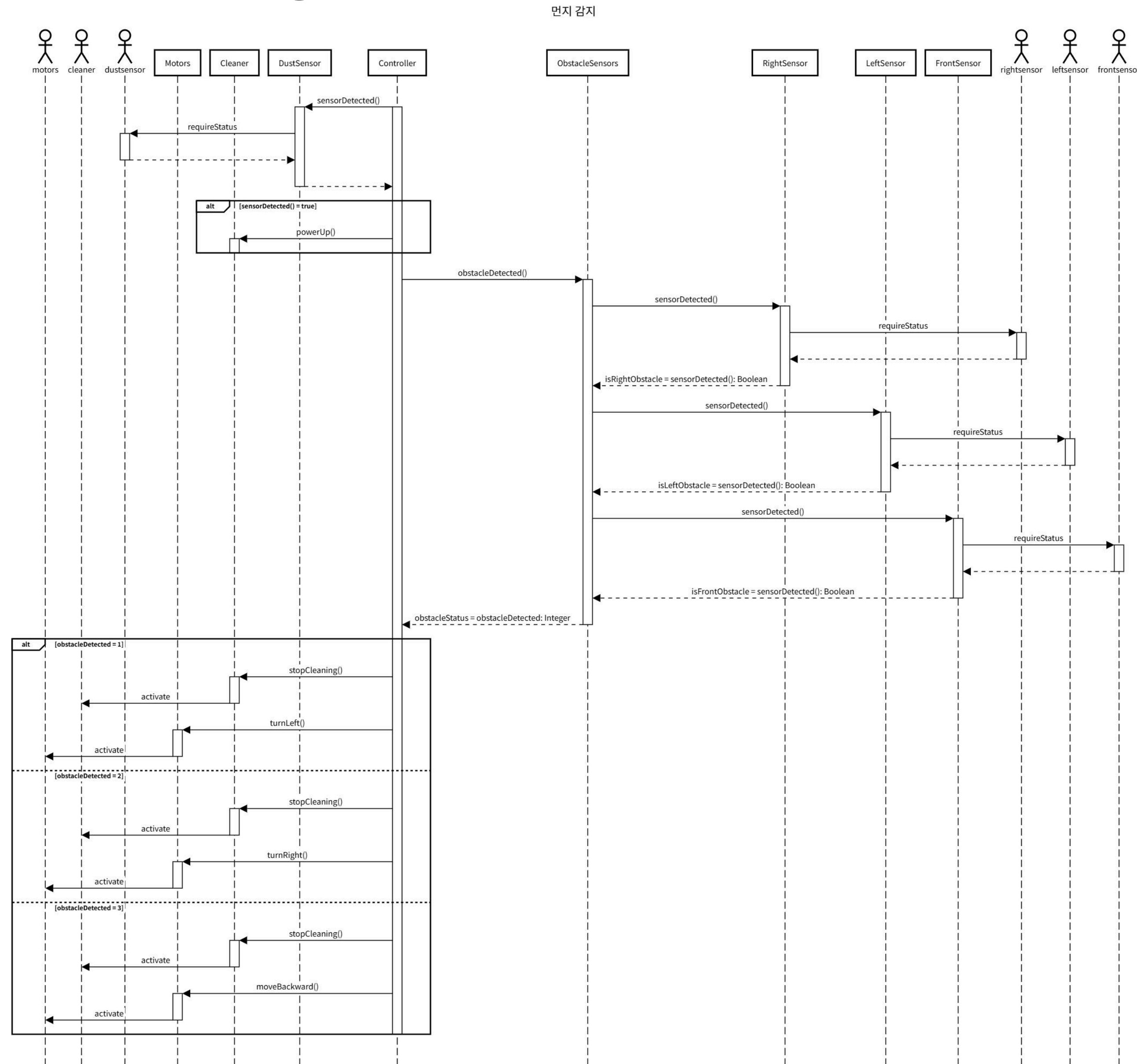
UC-07

최종 탈출 불가

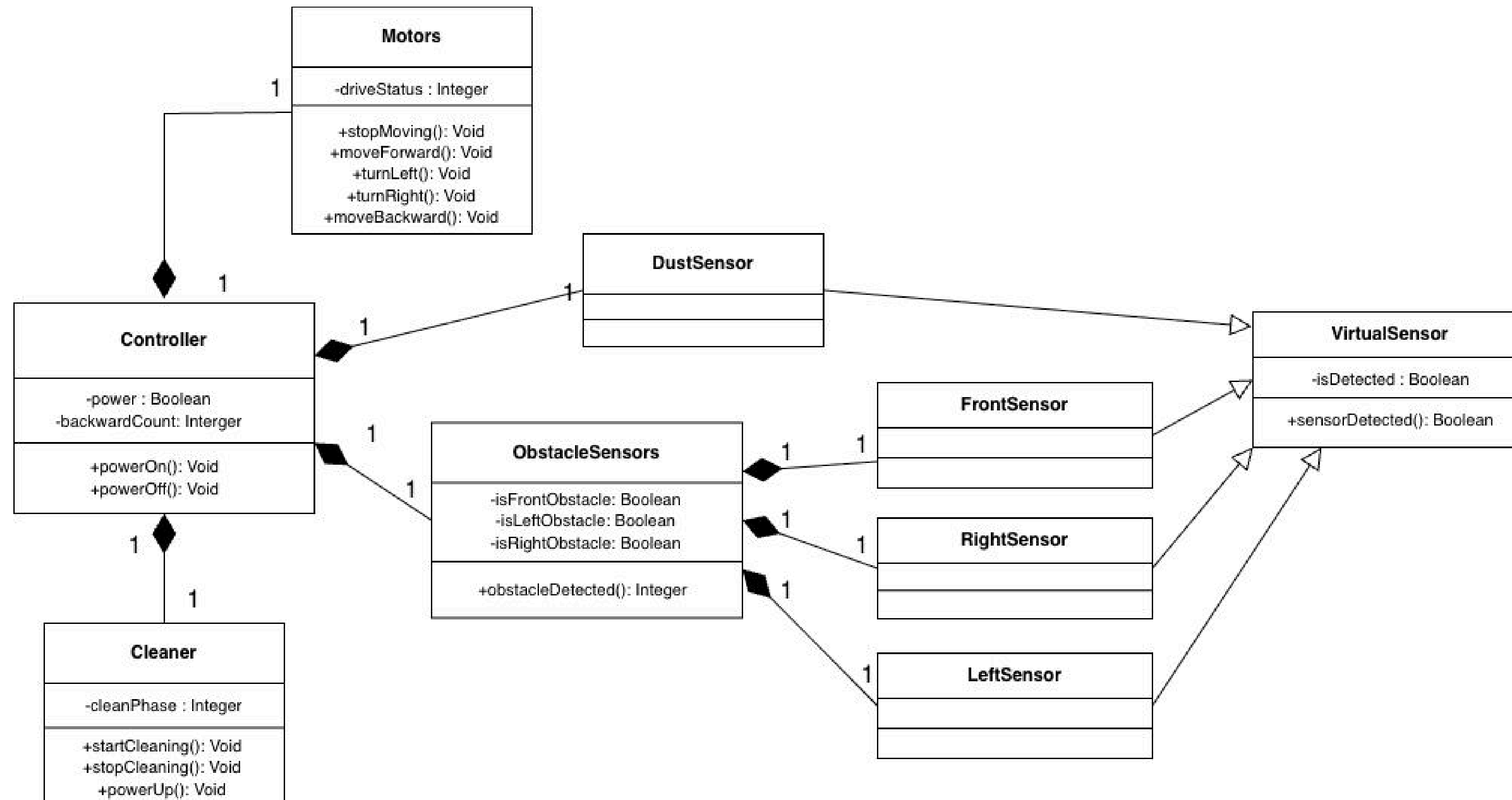


[OOD] Sequence Diagrams

UC-08



[OOD] Class Diagram



구현 (Coding)

Pair Programming

Pair Programming

이동훈 & 최환	엄정석 & 차민우
pre-commit, settings	CMakeLists, yamll
<ul style="list-style-type: none">• Controller.h• ObstacleSensors.h• SimulationData.h• Virtual Cleaner.h• Virtual Motors.h• VirtualSensor.h	<ul style="list-style-type: none">• test_cleaner.cpp• test_motors.cpp• test_obstacleSensors.cpp• test_sensors.cpp

구현 (Coding)

Unit Test

Unit Test

: MotorsTest

Constructor & Destructor Tests

```
TEST(MotorsTest, Constructor_test) {
    auto motors = std::make_unique<Motors>();
    EXPECT_EQ(motors->driveStatus, 0);
}

TEST(MotorsTest, Destructor_Test) {
    auto motors = std::make_unique<Motors>();
    motors->moveForward();
}
```

Function Tests

```
TEST(MotorsTest, moveForward_test) {
    auto motors = std::make_unique<Motors>();
    motors->moveForward();
    EXPECT_EQ(motors->driveStatus, 1);
}

TEST(MotorsTest, stopMoving_test) {
    auto motors = std::make_unique<Motors>();
    motors->stopMoving();
    EXPECT_EQ(motors->driveStatus, 0);
}

TEST(MotorsTest, turnLeft_test) {
    auto motors = std::make_unique<Motors>();
    motors->turnLeft();
    EXPECT_EQ(motors->driveStatus, 2);
}

TEST(MotorsTest, turnRight_test) {
    auto motors = std::make_unique<Motors>();
    motors->turnRight();
    EXPECT_EQ(motors->driveStatus, 3);
}

TEST(MotorsTest, moveBackward_test) {
    auto motors = std::make_unique<Motors>();
    motors->moveBackward();
    EXPECT_EQ(motors->driveStatus, -1);
}
```

Unit Test

: MotorsTest

SequentialTurn_test

```
TEST(MotorsTest, SequentialTurn_test) {  
    auto motors = std::make_unique<Motors>();  
  
    motors->turnLeft();  
    EXPECT_EQ(motors->driveStatus, 2);  
  
    motors->turnRight();  
    EXPECT_EQ(motors->driveStatus, 3);  
  
    motors->stopMoving();  
    EXPECT_EQ(motors->driveStatus, 0);  
}
```

FullMovement_test

```
TEST(MotorsTest, FullMovement_test) {  
    auto motors = std::make_unique<Motors>();  
  
    motors->moveForward();  
    EXPECT_EQ(motors->driveStatus, 1);  
  
    motors->moveBackward();  
    EXPECT_EQ(motors->driveStatus, -1);  
  
    motors->stopMoving();  
    EXPECT_EQ(motors->driveStatus, 0);  
}
```

Unit Test

: MotorsTest

RepeatAction_test

```
TEST(MotorsTest, RepeatAction_test) {
    auto motors = std::make_unique<Motors>();

    motors->moveForward();
    motors->moveForward();
    EXPECT_EQ(motors->driveStatus, 1);

    motors->stopMoving();
    EXPECT_EQ(motors->driveStatus, 0);
}
```

EmergencySwitch_test

```
TEST(MotorsTest, EmergencySwitch_test) {
    auto motors = std::make_unique<Motors>();

    motors->moveForward();
    motors->stopMoving();
    EXPECT_EQ(motors->driveStatus, 0);

    motors->moveBackward();
    EXPECT_EQ(motors->driveStatus, -1);
}
```

ZigZag_test

```
TEST(MotorsTest, ZigZag_test) {
    auto motors = std::make_unique<Motors>();

    motors->turnLeft();
    EXPECT_EQ(motors->driveStatus, 2);

    motors->turnRight();
    EXPECT_EQ(motors->driveStatus, 3);

    motors->turnLeft();
    EXPECT_EQ(motors->driveStatus, 2);

    motors->stopMoving();
    EXPECT_EQ(motors->driveStatus, 0);
}
```

Unit Test

: MotorsTest

SystemFullCycle_test

```
TEST(MotorsTest, SystemFullCycle_test) {
    auto motors = std::make_unique<Motors>();

    int statuses[] = {1, 2, 3, -1, 0};

    motors->moveForward();
    EXPECT_EQ(motors->driveStatus, statuses[0]);

    motors->turnLeft();
    EXPECT_EQ(motors->driveStatus, statuses[1]);

    motors->turnRight();
    EXPECT_EQ(motors->driveStatus, statuses[2]);

    motors->moveBackward();
    EXPECT_EQ(motors->driveStatus, statuses[3]);

    motors->stopMoving();
    EXPECT_EQ(motors->driveStatus, statuses[4]);
}
```

Unit Test

: CleanerTest

Constructor & Destructor Tests

```
TEST(CleanerTest, Constructor_test) {
    auto cleaner = std::make_unique<Cleaner>();
    EXPECT_EQ(cleaner->cleanPhase, 0);
}

TEST(CleanerTest, Destructor_Test) {
    auto cleaner = std::make_unique<Cleaner>();
    cleaner->startCleaning();
}
```

Function Tests

```
TEST(CleanerTest, startCleaning_test) {
    auto cleaner = std::make_unique<Cleaner>();
    cleaner->startCleaning();
    EXPECT_EQ(cleaner->cleanPhase, 1);
}

TEST(CleanerTest, stopCleaning_test) {
    auto cleaner = std::make_unique<Cleaner>();
    cleaner->startCleaning();
    cleaner->stopCleaning();
    EXPECT_EQ(cleaner->cleanPhase, 0);
}

TEST(CleanerTest, powerUp_test) {
    auto cleaner = std::make_unique<Cleaner>();
    cleaner->startCleaning();
    cleaner->powerUp();
    EXPECT_EQ(cleaner->cleanPhase, 2);
}
```

Unit Test

: CleanerTest

SequentialCleaning_test

```
TEST(CleanerTest, SequentialCleaning_test) {
    auto cleaner = std::make_unique<Cleaner>();

    cleaner->startCleaning();
    EXPECT_EQ(cleaner->cleanPhase, 1);

    cleaner->powerUp();
    EXPECT_EQ(cleaner->cleanPhase, 2);

    cleaner->stopCleaning();
    EXPECT_EQ(cleaner->cleanPhase, 0);
}
```

MultipleStart_test

```
TEST(CleanerTest, MultipleStart_test) {
    auto cleaner = std::make_unique<Cleaner>();

    cleaner->startCleaning();
    cleaner->startCleaning();
    EXPECT_EQ(cleaner->cleanPhase, 1);

    cleaner->stopCleaning();
    EXPECT_EQ(cleaner->cleanPhase, 0);
}
```

Unit Test

: CleanerTest

EmergencyStop_test

```
TEST(CleanerTest, EmergencyStop_test) {
    auto cleaner = std::make_unique<Cleaner>();

    cleaner->startCleaning();
    cleaner->stopCleaning();
    EXPECT_EQ(cleaner->cleanPhase, 0);

    cleaner->powerUp();
    EXPECT_EQ(cleaner->cleanPhase, 2);
}
```

FullCycle_test

```
TEST(CleanerTest, FullCycle_test) {
    auto cleaner = std::make_unique<Cleaner>();

    int phases[] = {1, 2, 0};

    cleaner->startCleaning();
    EXPECT_EQ(cleaner->cleanPhase, phases[0]);

    cleaner->powerUp();
    EXPECT_EQ(cleaner->cleanPhase, phases[1]);

    cleaner->stopCleaning();
    EXPECT_EQ(cleaner->cleanPhase, phases[2]);
}
```

Unit Test

: SensorTest - DustSensor

Function Tests

```
TEST(SensorTest, DustSensorTest_Default) {
    auto data = std::make_shared<SimulationData>();
    auto dustSensor = std::make_unique<DustSensor>(data);

    data->dust = false;
    EXPECT_FALSE(dustSensor->sensorDetected());
}

TEST(SensorTest, DustSensorTest_Detected) {
    auto data = std::make_shared<SimulationData>();
    auto dustSensor = std::make_unique<DustSensor>(data);

    data->dust = true;
    EXPECT_TRUE(dustSensor->sensorDetected());
}
```

Sequential Test

```
TEST(SensorTest, DustSensorTest_Sequential) {
    auto data = std::make_shared<SimulationData>();
    auto dustSensor = std::make_unique<DustSensor>(data);

    data->dust = false;
    EXPECT_FALSE(dustSensor->sensorDetected());

    data->dust = true;
    EXPECT_TRUE(dustSensor->sensorDetected());
}
```

Unit Test

: SensorTest - FrontSensor

Function Tests

```
TEST(SensorTest, FrontSensorTest_Default) {
    auto data = std::make_shared<SimulationData>();
    auto frontSensor = std::make_unique<FrontSensor>(data);

    data->frontObstacle = false;
    EXPECT_FALSE(frontSensor->sensorDetected());
}

TEST(SensorTest, FrontSensorTest_Detected) {
    auto data = std::make_shared<SimulationData>();
    auto frontSensor = std::make_unique<FrontSensor>(data);

    data->frontObstacle = true;
    EXPECT_TRUE(frontSensor->sensorDetected());
}
```

Sequential Test

```
TEST(SensorTest, FrontSensorTest_Sequential) {
    auto data = std::make_shared<SimulationData>();
    auto frontSensor = std::make_unique<FrontSensor>(data);

    data->frontObstacle = false;
    EXPECT_FALSE(frontSensor->sensorDetected());

    data->frontObstacle = true;
    EXPECT_TRUE(frontSensor->sensorDetected());
}
```

Unit Test

: SensorTest - LeftSensor

Function Tests

```
TEST(SensorTest, LeftSensorTest_Default) {
    auto data = std::make_shared<SimulationData>();
    auto leftSensor = std::make_unique<LeftSensor>(data);

    data->leftObstacle = false;
    EXPECT_FALSE(leftSensor->sensorDetected());
}

TEST(SensorTest, LeftSensorTest_Detected) {
    auto data = std::make_shared<SimulationData>();
    auto leftSensor = std::make_unique<LeftSensor>(data);

    data->leftObstacle = true;
    EXPECT_TRUE(leftSensor->sensorDetected());
}
```

Sequential Test

```
TEST(SensorTest, LeftSensorTest_Sequential) {
    auto data = std::make_shared<SimulationData>();
    auto leftSensor = std::make_unique<LeftSensor>(data);

    data->leftObstacle = false;
    EXPECT_FALSE(leftSensor->sensorDetected());

    data->leftObstacle = true;
    EXPECT_TRUE(leftSensor->sensorDetected());
}
```

Unit Test

: SensorTest - RightSensor

Function Tests

```
TEST(SensorTest, RightSensorTest_Default) {
    auto data = std::make_shared<SimulationData>();
    auto rightSensor = std::make_unique<RightSensor>(data);

    data->rightObstacle = false;
    EXPECT_FALSE(rightSensor->sensorDetected());
}

TEST(SensorTest, RightSensorTest_Detected) {
    auto data = std::make_shared<SimulationData>();
    auto rightSensor = std::make_unique<RightSensor>(data);

    data->rightObstacle = true;
    EXPECT_TRUE(rightSensor->sensorDetected());
}
```

Sequential Test

```
TEST(SensorTest, RightSensorTest_Sequential) {
    auto data = std::make_shared<SimulationData>();
    auto rightSensor = std::make_unique<RightSensor>(data);

    data->rightObstacle = false;
    EXPECT_FALSE(rightSensor->sensorDetected());

    data->rightObstacle = true;
    EXPECT_TRUE(rightSensor->sensorDetected());
}
```

Unit Test

: ObstacleSensorsTest

Default

```
TEST(ObstacleSensorsTest, Default) {
    auto data = std::make_shared<SimulationData>();
    auto frontSensor = std::make_shared<FrontSensor>(data);
    auto rightSensor = std::make_shared<RightSensor>(data);
    auto leftSensor = std::make_shared<LeftSensor>(data);

    auto obstacleSensors = std::make_shared<ObstacleSensors>(frontSensor, leftSensor, rightSensor);

    EXPECT_EQ(obstacleSensors->obstacleDetected(), 0);
}
```

Front

```
TEST(ObstacleSensorsTest, Front) {
    auto data = std::make_shared<SimulationData>();
    auto frontSensor = std::make_shared<FrontSensor>(data);
    auto rightSensor = std::make_shared<RightSensor>(data);
    auto leftSensor = std::make_shared<LeftSensor>(data);

    auto obstacleSensors = std::make_shared<ObstacleSensors>(frontSensor, leftSensor, rightSensor);

    data->frontObstacle = true;

    EXPECT_EQ(obstacleSensors->obstacleDetected(), 1);
}
```

Unit Test

: ObstacleSensorsTest

Left

```
TEST(ObstacleSensorsTest, Left) {
    auto data = std::make_shared<SimulationData>();
    auto frontSensor = std::make_shared<FrontSensor>(data);
    auto rightSensor = std::make_shared<RightSensor>(data);
    auto leftSensor = std::make_shared<LeftSensor>(data);

    auto obstacleSensors = std::make_shared<ObstacleSensors>(frontSensor, leftSensor, rightSensor);

    data->leftObstacle = true;

    EXPECT_EQ(obstacleSensors->obstacleDetected(), 0);
}
```

Right

```
TEST(ObstacleSensorsTest, Right) {
    auto data = std::make_shared<SimulationData>();
    auto frontSensor = std::make_shared<FrontSensor>(data);
    auto rightSensor = std::make_shared<RightSensor>(data);
    auto leftSensor = std::make_shared<LeftSensor>(data);

    auto obstacleSensors = std::make_shared<ObstacleSensors>(frontSensor, leftSensor, rightSensor);

    data->rightObstacle = true;

    EXPECT_EQ(obstacleSensors->obstacleDetected(), 0);
}
```

Unit Test

: ObstacleSensorsTest

Front_Left

```
TEST(ObstacleSensorsTest, Front_Left) {
    auto data = std::make_shared<SimulationData>();
    auto frontSensor = std::make_shared<FrontSensor>(data);
    auto rightSensor = std::make_shared<RightSensor>(data);
    auto leftSensor = std::make_shared<LeftSensor>(data);

    auto obstacleSensors = std::make_shared<ObstacleSensors>(frontSensor, leftSensor, rightSensor);

    data->frontObstacle = true;
    data->leftObstacle = true;

    EXPECT_EQ(obstacleSensors->obstacleDetected(), 2);
}
```

Front_Right

```
TEST(ObstacleSensorsTest, Front_Right) {
    auto data = std::make_shared<SimulationData>();
    auto frontSensor = std::make_shared<FrontSensor>(data);
    auto rightSensor = std::make_shared<RightSensor>(data);
    auto leftSensor = std::make_shared<LeftSensor>(data);

    auto obstacleSensors = std::make_shared<ObstacleSensors>(frontSensor, leftSensor, rightSensor);

    data->frontObstacle = true;
    data->rightObstacle = true;

    EXPECT_EQ(obstacleSensors->obstacleDetected(), 3);
}
```

Unit Test

: ObstacleSensorsTest

Front_Left_Right

```
TEST(ObstacleSensorsTest, Front_Left_Right) {
    auto data = std::make_shared<SimulationData>();
    auto frontSensor = std::make_shared<FrontSensor>(data);
    auto rightSensor = std::make_shared<RightSensor>(data);
    auto leftSensor = std::make_shared<LeftSensor>(data);

    auto obstacleSensors = std::make_shared<ObstacleSensors>(frontSensor, leftSensor, rightSensor);

    data->frontObstacle = true;
    data->leftObstacle = true;
    data->rightObstacle = true;

    EXPECT_EQ(obstacleSensors->obstacleDetected(), 4);
}
```

Front_Right

```
TEST(ObstacleSensorsTest, Left_Right) {
    auto data = std::make_shared<SimulationData>();
    auto frontSensor = std::make_shared<FrontSensor>(data);
    auto rightSensor = std::make_shared<RightSensor>(data);
    auto leftSensor = std::make_shared<LeftSensor>(data);

    auto obstacleSensors = std::make_shared<ObstacleSensors>(frontSensor, leftSensor, rightSensor);

    data->leftObstacle = true;
    data->rightObstacle = true;

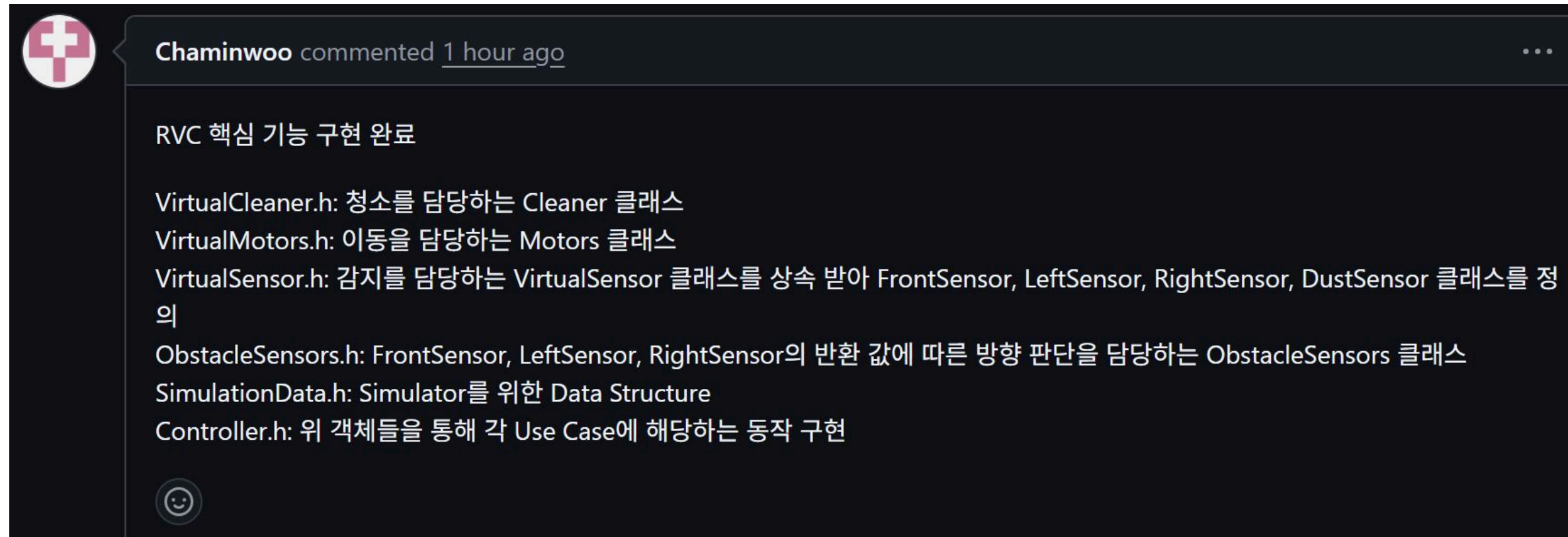
    EXPECT_EQ(obstacleSensors->obstacleDetected(), 5);
}
```

구현 (Coding)

Code Review

Code Review

: main

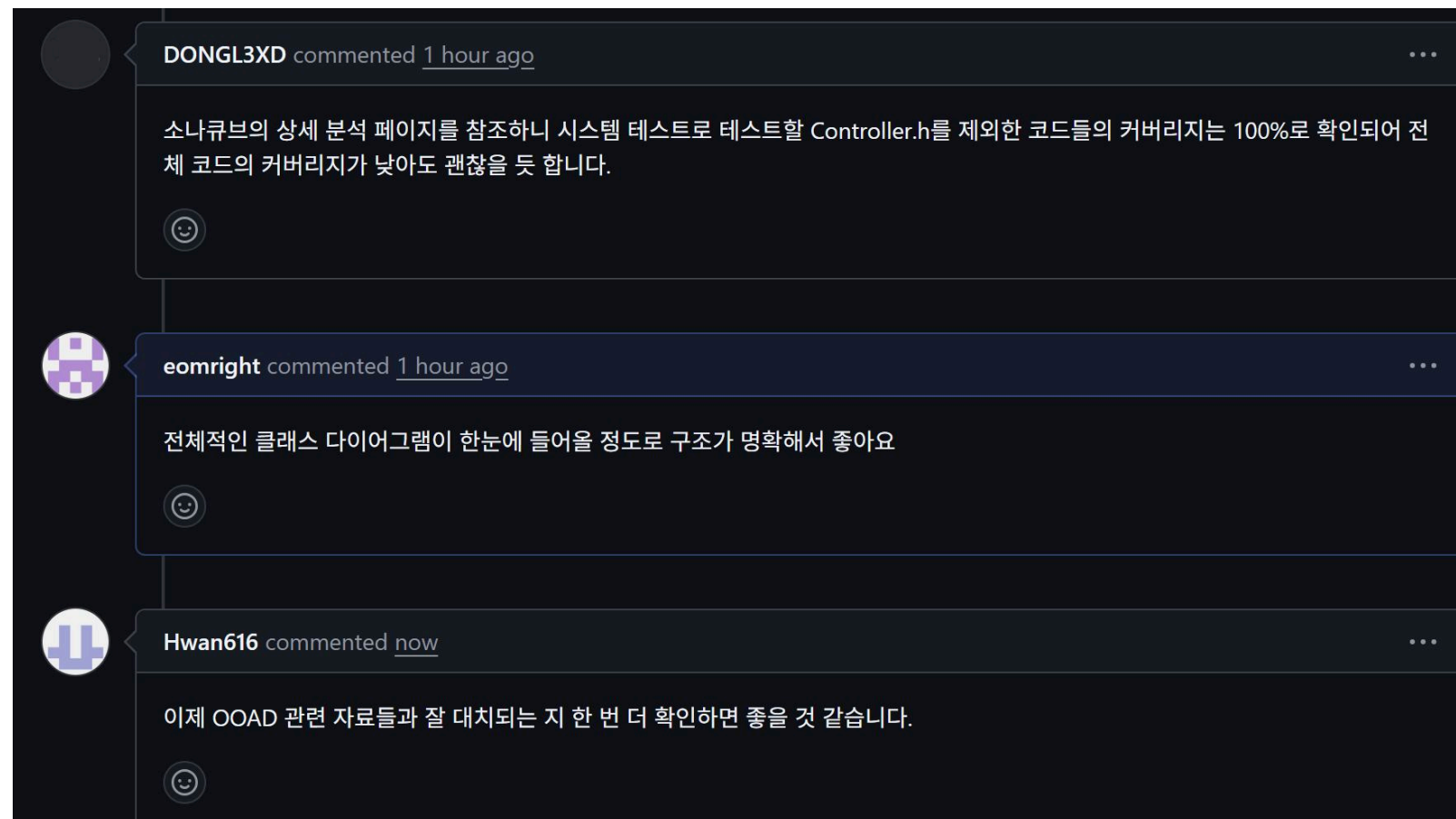


Chaminwoo commented [1 hour ago](#)

RVC 핵심 기능 구현 완료

- VirtualCleaner.h: 청소를 담당하는 Cleaner 클래스
- VirtualMotors.h: 이동을 담당하는 Motors 클래스
- VirtualSensor.h: 감지를 담당하는 VirtualSensor 클래스를 상속 받아 FrontSensor, LeftSensor, RightSensor, DustSensor 클래스를 정의
- ObstacleSensors.h: FrontSensor, LeftSensor, RightSensor의 반환 값에 따른 방향 판단을 담당하는 ObstacleSensors 클래스
- SimulationData.h: Simulator를 위한 Data Structure
- Controller.h: 위 객체들을 통해 각 Use Case에 해당하는 동작 구현

PR Description



DONGL3XD commented [1 hour ago](#)

소나큐브의 상세 분석 페이지를 참조하니 시스템 테스트로 테스트할 Controller.h를 제외한 코드들의 커버리지는 100%로 확인되어 전체 코드의 커버리지가 낮아도 괜찮을 듯 합니다.

eomright commented [1 hour ago](#)

전체적인 클래스 다이어그램이 한눈에 들어올 정도로 구조가 명확해서 좋아요

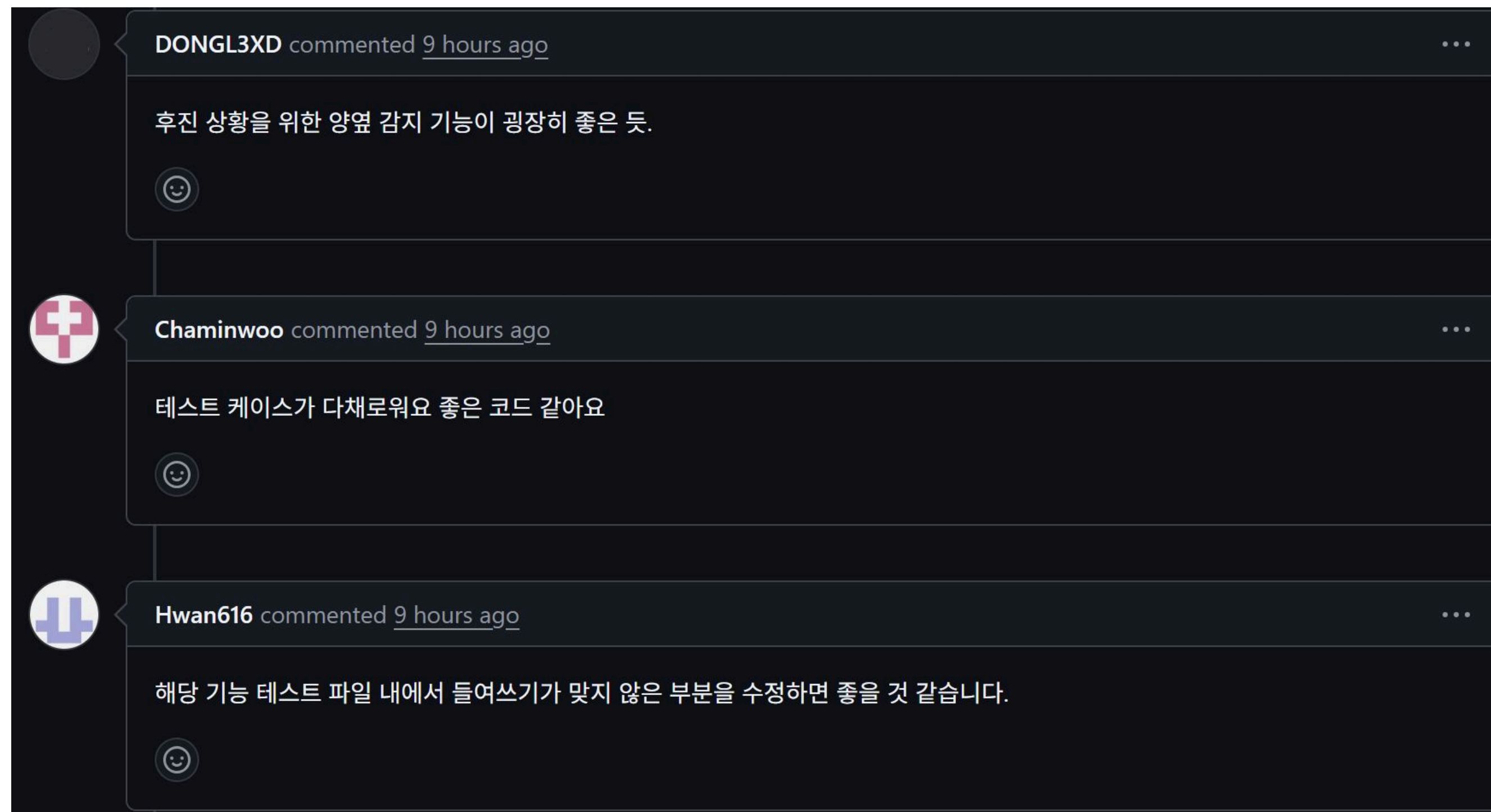
Hwan616 commented [now](#)

이제 OOAD 관련 자료들과 잘 대치되는 지 한 번 더 확인하면 좋을 것 같습니다.

Review

Code Review

: Sensors




Review


Code Review

: API call

PR Description

 **DONGL3XD** commented 4 hours ago

HW API Call이 들어갈 부분을 명시하는 기능 추가



Review

eomright commented 3 hours ago ...

Negative 테스트로 '모터 고장'이나 '감지 실패'도 체크해야 해서 각 클래스에 강제로 에러를 발생시키는 메서드를 미리 넣어두면 좋을 것 같음



Hwan616 commented 3 hours ago ...

Negative 테스트를 위한거면, 어차피 각 클래스에서 변수들이 public으로 선언되어 있으니까, 시뮬레이터 안에서 에러 발생 함수 만들어도 괜찮을 것 같습니다.



Chaminwoo commented 2 hours ago ...

API 명시를 추가해서 하드웨어까지 구현하지 않더라도 좀 더 명확한 코드 구조와 하드웨어 연결을 보장하겠네요!



System Test

Simulator

Simulator

```
----- [Simulator] RVC Status -----  
  
          OBSTL  
          |RVC|  
  
----- [Simulator] 장애물/면지 상태 입력 -----  
- 0: 아무것도 감지되지 않음  
- 1: 정면만 감지  
- 2: 정면, 좌측 감지  
- 3: 정면, 우측 감지  
- 4: 정면, 좌측, 우측 감지  
- 5: 좌측, 우측 감지 (후진용)  
- 6: 면지 감지되지 않음  
- 7: 면지 감지  
- 8: 전원 켜기/끄기  
- 9: 시뮬레이터 종료  
  
- 11: 정면 장애물 센서 에러  
- 12: 좌측 장애물 센서 에러  
- 13: 우측 장애물 센서 에러  
- 14: 면지 센서 에러  
- 15: 클리너 에러  
- 16: 모터 에러  
|
```

System Test

System Test Cases

System Test Cases

positive	negative
전원 켜기	전원 켜는 중 클리너 오작동
전원 끄기	전원 켜는 중 전체 장애물 센서 고장
기본 주행	먼지 흡입 후 power up 유지
전방 장애물 감지	최종 탈출 불가
전방, 좌측 장애물 감지	전방 장애물 감지 실패
전방, 우측 장애물 감지	좌측 장애물 감지 실패 (논리상 우회전 불가로 좌회전)
전방, 좌우측 장애물 감지	모터 에러 (장애물 감지에도 이동 불능)

System Test Cases

positive	negative
먼지 감지	좌회전 시작 후 우측 센서 고장
후진 중 탈출	우회전 시작 후 cleaner 고장으로 청소 재개 불가
전방 장애물 감지 및 먼지 감지	후진 중 좌,우측 센서 고장으로 전진
전방, 좌측 장애물 감지 및 먼지 감지	주행 중 청소 불가
전방, 우측 장애물 감지 및 먼지 감지	회전 후 청소 불가
전방, 좌우측 장애물 감지 및 먼지 감지	
후진 후 우측 장애물 감지	

System Test Cases

Positive	Negative
후진 후 좌측 장애물 감지	
후진 후 장애물 감지하지 않을 경우	
좌회전 시작 후 센서 값이 “전방+좌측”으로 바뀌어도 중단 없이 회전	
우회전 시작 후 센서 값이 “전방+우측”으로 바뀌어도 중단 없이 회전	

System Test

Test 수행 과정 및 결과

Test 수행 과정 및 결과

19. 전원 켜는 중 클리너 오작동으로 전진만 수행

```

matches selected pattern: *
[info] Module: C:\Windows\System32\ucrtbased.dll is selected because it matches selected pattern: *
[info] Module: C:\Windows\System32\vcruntime140_1d.dll is selected because it matches selected pattern: *
[Controller]시스템 시작
[Cleaner]청소 LV-1
[Motors]전진
[Motors]전진
[Cleaner]청소 LV-1
[Cleaner]FAIL: 에러 발생
[Motors]전진
[Cleaner]FAIL: 에러로 인한 작동 불가
[Cleaner]FAIL: 에러 발생
[Motors]전진
[Cleaner]FAIL: 에러로 인한 작동 불가
[Cleaner]FAIL: 에러 발생
[Motors]전진
[Cleaner]FAIL: 에러로 인한 작동 불가
[Cleaner]FAIL: 에러 발생
[Motors]전진
[Cleaner]FAIL: 에러로 인한 작동 불가
[Cleaner]FAIL: 에러 발생
[Motors]전진
[Cleaner]FAIL: 에러로 인한 작동 불가
[Cleaner]FAIL: 에러 발생
[Motors]전진
[Cleaner]FAIL: 에러로 인한 작동 불가
[Cleaner]FAIL: 에러 발생
[Motors]정지
[Controller]시스템 종료

----- [Simulator] 장애물/면지 상태 입력 -----
- 0: 아무것도 감지되지 않음
- 1: 정면만 감지
- 2: 정면, 좌측 감지
- 3: 정면, 우측 감지
- 4: 정면, 좌측, 우측 감지
- 5: 좌측, 우측 감지 (후진용)
- 6: 면지 감지되지 않음
- 7: 면지 감지
- 8: 전원 켜기/끄기
- 9: 시뮬레이터 종료

- 11: 정면 장애물 센서 에러
- 12: 좌측 장애물 센서 에러
- 13: 우측 장애물 센서 에러
- 14: 면지 센서 에러
- 15: 클리너 에러
- 16: 모터 에러

[Simulator] RVC Software 전원을 끕니다.

----- [Simulator] 장애물/면지 상태 입력 -----
- 0: 아무것도 감지되지 않음
- 1: 정면만 감지
- 2: 정면, 좌측 감지
- 3: 정면, 우측 감지
- 4: 정면, 좌측, 우측 감지
- 5: 좌측, 우측 감지 (후진용)
- 6: 면지 감지되지 않음
- 7: 면지 감지
- 8: 전원 켜기/끄기
- 9: 시뮬레이터 종료

- 11: 정면 장애물 센서 에러
- 12: 좌측 장애물 센서 에러
- 13: 우측 장애물 센서 에러
- 14: 면지 센서 에러
- 15: 클리너 에러
- 16: 모터 에러

Windows 정품 인증
[설정]으로 이동하여 Windows를 정품 인증합니다.
    
```

20. 전원 켜는 중 전체 장애물 센서 고장으로 전진만 수행

```

[info] Module: C:\Windows\System32\msvcpl140d.dll is selected because it matches selected pattern: *
[info] Module: C:\Windows\System32\vcruntime140_1d.dll is selected because it matches selected pattern: *
[info] Module: C:\Windows\System32\ucrtbased.dll is selected because it matches selected pattern: *
[info] Module: C:\Windows\System32\ucrtbased.dll is selected because it matches selected pattern: *
[info] Module: C:\Windows\System32\ucrtbased.dll is selected because it matches selected pattern: *
[Controller]시스템 시작
[Cleaner]청소 LV-1
[Motors]전진
[FrontSensor]FAIL: 에러 발생
[LeftSensor]FAIL: 에러 발생
[RightSensor]FAIL: 에러 발생
[FrontSensor]FAIL: 에러로 인한 작동 불가
[LeftSensor]FAIL: 에러로 인한 작동 불가
[RightSensor]FAIL: 에러로 인한 작동 불가
[Motors]전진
[Cleaner]청소 LV-1
[FrontSensor]FAIL: 에러 발생
[LeftSensor]FAIL: 에러 발생
[RightSensor]FAIL: 에러 발생
[FrontSensor]FAIL: 에러로 인한 작동 불가
[LeftSensor]FAIL: 에러로 인한 작동 불가
[RightSensor]FAIL: 에러로 인한 작동 불가
[Motors]전진
[Cleaner]청소 LV-1
[FrontSensor]FAIL: 에러 발생
[LeftSensor]FAIL: 에러 발생
[RightSensor]FAIL: 에러 발생
[FrontSensor]FAIL: 에러로 인한 작동 불가
[LeftSensor]FAIL: 에러로 인한 작동 불가
[RightSensor]FAIL: 에러로 인한 작동 불가
[Motors]정지
[Controller]시스템 종료

----- [Simulator] 장애물/면지 상태 입력 -----
- 0: 아무것도 감지되지 않음
- 1: 정면만 감지
- 2: 정면, 좌측 감지
- 3: 정면, 우측 감지
- 4: 정면, 좌측, 우측 감지
- 5: 좌측, 우측 감지 (후진용)
- 6: 면지 감지되지 않음
- 7: 면지 감지
- 8: 전원 켜기/끄기
- 9: 시뮬레이터 종료

- 11: 정면 장애물 센서 에러
- 12: 좌측 장애물 센서 에러
- 13: 우측 장애물 센서 에러
- 14: 면지 센서 에러
- 15: 클리너 에러
- 16: 모터 에러

[Simulator] RVC Software 전원을 끕니다.

----- [Simulator] 장애물/면지 상태 입력 -----
- 0: 아무것도 감지되지 않음
- 1: 정면만 감지
- 2: 정면, 좌측 감지
- 3: 정면, 우측 감지
- 4: 정면, 좌측, 우측 감지
- 5: 좌측, 우측 감지 (후진용)
- 6: 면지 감지되지 않음
- 7: 면지 감지
- 8: 전원 켜기/끄기
- 9: 시뮬레이터 종료



- 11: 정면 장애물 센서 에러
- 12: 좌측 장애물 센서 에러
- 13: 우측 장애물 센서 에러
- 14: 면지 센서 에러
- 15: 클리너 에러
- 16: 모터 에러

Windows 정품 인증
[설정]으로 이동하여 Windows를 정품 인증합니다.
    
```









Static Code Analysis

정적분석 수행

정적분석 수행

  Chaminwoo merged commit 53d2242 into `main` 20 minutes ago Hide details Revert

7 checks passed

✓  Style Check (cppLint)	Details
✓  Style Check (cppLint)	Details
✓  Static Analysis (cppCheck)	Details
✓  Static Analysis (cppCheck)	Details
✓  Build and Scan (gTest, SonaQubeCloud)	Details
✓  Build and Scan (gTest, SonaQubeCloud)	Details
✓  SonarCloud Code Analysis Quality Gate passed	Details

< cppLint & cppCheck & SonaQubeCloud >

Static Code Analysis

수행 결과

수행 결과

OOAD-Team9 / RVC_Controller / Summary

Summary

main

Private • 350 Lines of Code • Last analysis 14 minutes ago • [53d22427](#)[View on GitHub](#)

Last analysis had a warning

[View warning](#) Quality Gate: [Sonar way](#)
Passed The quality profile assigned to this project has changed. You can explore the change in the [Activity page](#).

New Code

Overall Code

New code Since 8 days ago

New Issues

6

No conditions set



Accepted Issues

0

Valid issues that were not fixed



Coverage

98.1%Required: $\geq 80.0\%$
on **108** New Lines to cover

Duplications

0.0%Required: $\leq 3.0\%$
on **933** New Lines

Security Hotspots

0

No conditions set

수행 결과

OOAD-Team9 / RVC_Controller / Measures

Measures main ✓

Coverage	98.1%
Lines to Cover	108
Uncovered Lines	0
Line Coverage	100%
Conditions to Cover	50
Uncovered Conditions	3
Condition Coverage	94.0%
Overall Code	
Coverage	98.1%
Lines to Cover	108
Uncovered Lines	0
Line Coverage	100%
Conditions to Cover	50
Uncovered Conditions	3
Condition Coverage	94.0%

RVC_Controller > include 📄 View as Tree 6 files

Coverage 98.1% New code: last 30 days

	Coverage	Uncovered Lines	Uncovered Conditions
📄 Controller.h	-	-	-
📄 ObstacleSensors.h	100%	0	0
📄 SimulationData.h	-	-	-
📄 VirtualCleaner.h	100%	0	0
📄 VirtualMotors.h	94.4%	0	3
📄 VirtualSensor.h	100%	0	0

6 of 6 shown

© 2018-2026 [SonarSource Sàrl](#). All rights reserved. [Terms](#) [Pricing](#) [Privacy](#) [Cookie Policy](#) [Security](#) [Community](#) [Documentation](#) [Contact us](#) [Status](#) [About](#)

수행 결과

OOAD-Team9 / RVC_Controller / Code

Code

main ▾

[Open in Architecture](#)

	Lines of Code	Security	Reliability	Maintainability	Security Hotspots	Coverage	Duplications
📁 RVC_Controller							
📁 include	328	0	0	6	0	98.1%	0.0%
📁 src	22	0	0	0	0	—	0.0%
📁 tests	—	0	0	0	0	—	—

3 of 3 shown

감사합니다.