Sample Result

OUTPUT 결과는 다음과 같다.

```
(jh) kimjoowhan@gimjuhwan-ui-MacBookPro visitor_JoowhanKim % java Main.java
=E: Systen/Component Checking on Gasoline Engine
Engine Oil To Be Checked on Gasoline Engine
Gasoline Engine Log/Observer: Suggesting Oil Change

=M: Systen/Component Checking on Electric Motor
Can Vibration with Electric Motor
Electric Motor Log/Observer: Suggesting Motor Mounting Bolt Checking

=B: Systen/Component Checking on Regenerative Brake
Abnormal Speed Decrease with Regenerative Brake
Suggesting Brake and Generator Gear Checking


PHEV Checkingus log.html is made

==E: Self Repairing on Gasoline Engine
Exchange oil for Gasoline Engine
Gasoline Engine Log/Observer: Reparing engine oil warning

==M: Self Repairing on Electric Motor
Turn Tightly Motor Mounting Bolt of Electric Motor
Electric Motor Log/Observer: Repairing Car Vibration with Motor

==B: Self Repairing on Regenerative Brake
Monitoring Brake and Generator Gear of Regenerative Brake
Regenerative Brake System Log/Observer: Repairing Abnormal Speed Decrease with Regenerative Brake System


===E: Software Control and Recovery for Gasoline Engine
Increase the control software level of oil supply for Gasoline Engine
Gasoline Engine Log/Observer: Resetting Oil Level for engine oil warning

===M: Software Control and Recovery for Electric Motor
Initiate Monitoring SW for Motor Vibration of  Electric Motor
Electric Motor Log/Observer: Resetting the Monitoring SW for Car Vibration with Motor Action

===B: Software Control and Recovery for Regenerative Brake
Initiate Speed monitoring SW for Regenerative Brake
Regenerative Brake System Log/Observer: Resetting Speed Monitoring SW with Regenerative Brake System
    Engine Status: oil
    Engine Status: vibratingMotor
    Engine Status: abnormalSpeedDecrease
    Run with Stop Mode
```

```
(jh) kimjoowhan@gimjuhwan-ui-MacBookPro visitor_JoowhanKim % javac *.java
(jh) kimjoowhan@gimjuhwan-ui-MacBookPro visitor_JoowhanKim % java Main.java
=E: Systen/Component Checking on Gasoline Engine
Engine Oil To Be Checked on Gasoline Engine
Gasoline Engine Log/Observer: Suggesting Oil Change

=M: Systen/Component Checking on Electric Motor
Can Vibration with Electric Motor
Electric Motor Log/Observer: Suggesting Motor Mounting Bolt Checking

=B: Systen/Component Checking on Regenerative Brake
Normal Regen. Brake Working with Regenerative Brake
Regen. Brake Working OK


PHEV Checkingus log.html is made

==E: Self Repairing on Gasoline Engine
Exchange oil for Gasoline Engine
Gasoline Engine Log/Observer: Reparing engine oil warning

==M: Self Repairing on Electric Motor
Turn Tightly Motor Mounting Bolt of Electric Motor
Electric Motor Log/Observer: Repairing Car Vibration with Motor

==B: Self Repairing on Regenerative Brake
Regenerative Brake System Log/Observer: Regenerative Brake System OK


===E: Software Control and Recovery for Gasoline Engine
Increase the control software level of oil supply for Gasoline Engine
Gasoline Engine Log/Observer: Resetting Oil Level for engine oil warning

===M: Software Control and Recovery for Electric Motor
Initiate Monitoring SW for Motor Vibration of  Electric Motor
Electric Motor Log/Observer: Resetting the Monitoring SW for Car Vibration with Motor Action

===B: Software Control and Recovery for Regenerative Brake
Regenerative Brake System Log/Observer: Keep Current SW Monitoring Module for Regenerative Brake System
    Engine Status: oil
    Engine Status: vibratingMotor
    Engine Status:
    Run with Engine Running with Slow-Down Mode
(jh) kimjoowhan@gimjuhwan-ui-MacBookPro visitor_JoowhanKim % 
```

```
(jh) kimjoowhan@gimjuhwan-ui-MacBookPro visitor_JoowhanKim % javac *.java
(jh) kimjoowhan@gimjuhwan-ui-MacBookPro visitor_JoowhanKim % java VisitorDemo.java
=E: Systen/Component Checking on Gasoline Engine
Gasoline Engine Log/Observer: Engine Running OK

=M: Systen/Component Checking on Electric Motor
Normal Motor Power
Electric Motor Log/Observer: Motor Running OK

=B: Systen/Component Checking on Regenerative Brake
Normal Regen. Brake Working with Regenerative Brake
Regen. Brake Working OK


PHEV Checkingus log.html is made

==E: Self Repairing on Gasoline Engine
Gasoline Engine Log/Observer: EngineStatus Good

==M: Self Repairing on Electric Motor
Electric Motor Log/Observer: Motor status OK

==B: Self Repairing on Regenerative Brake
Regenerative Brake System Log/Observer: Regenerative Brake System OK


===E: Software Control and Recovery for Gasoline Engine
Gasoline Engine Log/Observer: Keep Curren SW Monitoring MOdule for Engine

===M: Software Control and Recovery for Electric Motor
Electric Motor Log/Observer: Keep Curren SW Monitoring Module for Motor

===B: Software Control and Recovery for Regenerative Brake
Regenerative Brake System Log/Observer: Keep Current SW Monitoring Module for Regenerative Brake System
    Engine Status: OK
    Engine Status: OK
    Engine Status:
    Run with Engine Running with Slow-Down Mode
(jh) kimjoowhan@gimjuhwan-ui-MacBookPro visitor_JoowhanKim % █
```

```
(jh) kimjoowhan@gimjuhwan-ui-MacBookPro visitor_JoowhanKim % javac *.java
(jh) kimjoowhan@gimjuhwan-ui-MacBookPro visitor_JoowhanKim % java VisitorDemo.java
=E: Systen/Component Checking on Gasoline Engine
Gasoline Engine Log/Observer: Engine Running OK

=M: Systen/Component Checking on Electric Motor
Can Vibration with Electric Motor
Electric Motor Log/Observer: Suggesting Motor Mounting Bolt Checking

=B: Systen/Component Checking on Regenerative Brake
Abnormal Speed Decrease with Regenerative Brake
Suggesting Brake and Generator Gear Checking


PHEV Checkingus log.html is made

==E: Self Repairing on Gasoline Engine
Gasoline Engine Log/Observer: EngineStatus Good

==M: Self Repairing on Electric Motor
Turn Tightly Motor Mounting Bolt of Electric Motor
Electric Motor Log/Observer: Repairing Car Vibration with Motor

==B: Self Repairing on Regenerative Brake
Monitoring Brake and Generator Gear of Regenerative Brake
Regenerative Brake System Log/Observer: Repairing Abnormal Speed Decrease with Regenerative Brake System


===E: Software Control and Recovery for Gasoline Engine
Gasoline Engine Log/Observer: Keep Curren SW Monitoring MOdule for Engine

===M: Software Control and Recovery for Electric Motor
Initiate Monitoring SW for Motor Vibration of  Electric Motor
Electric Motor Log/Observer: Resetting the Monitoring SW for Car Vibration with Motor Action

===B: Software Control and Recovery for Regenerative Brake
Initiate Speed monitoring SW for Regenerative Brake
Regenerative Brake System Log/Observer: Resetting Speed Monitoring SW with Regenerative Brake System
    Engine Status: OK
    Engine Status: vibratingMotor
    Engine Status: abnormalSpeedDecrease
    Run with Engine Mode
(jh) kimjoowhan@gimjuhwan-ui-MacBookPro visitor_JoowhanKim %
```
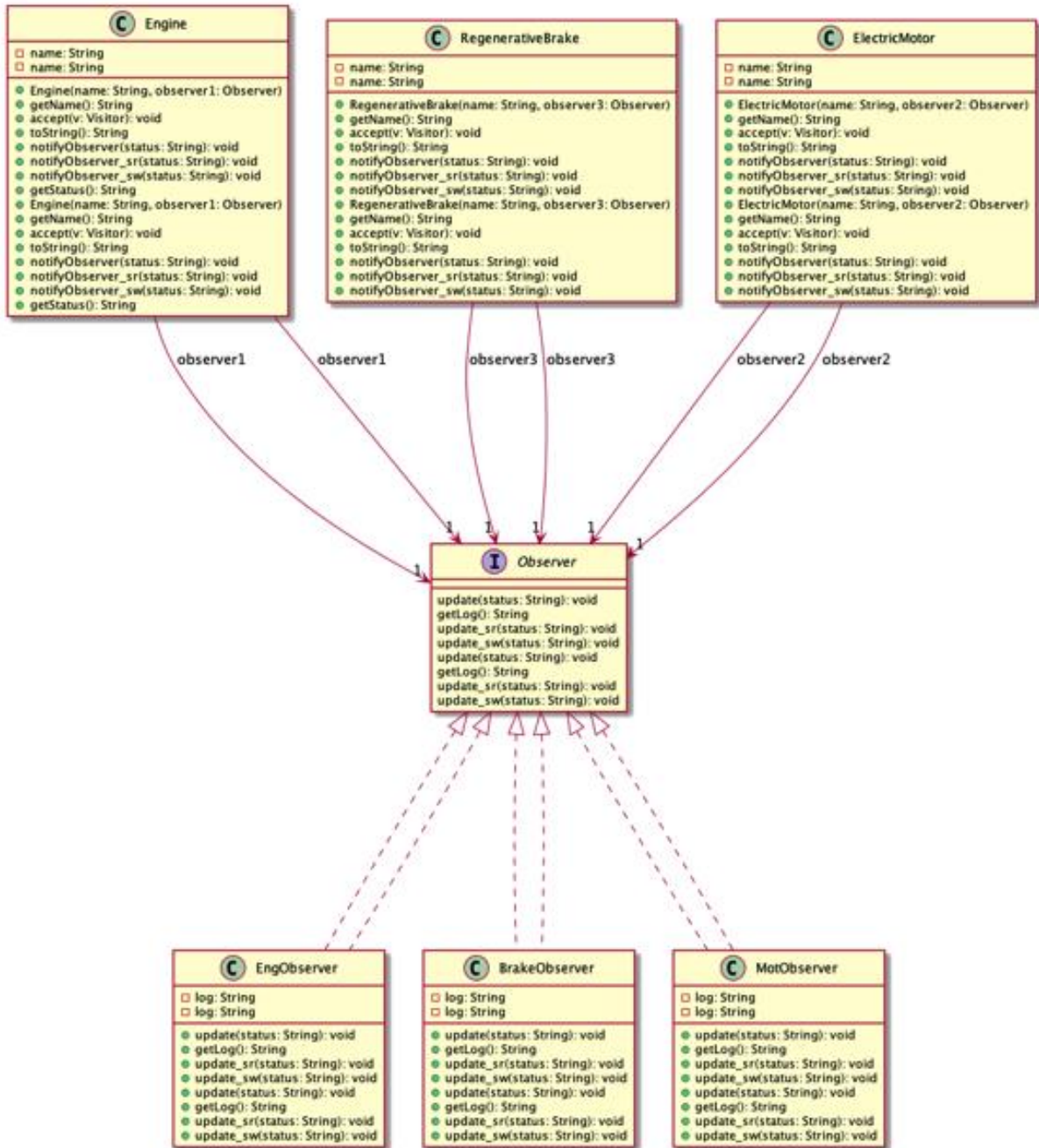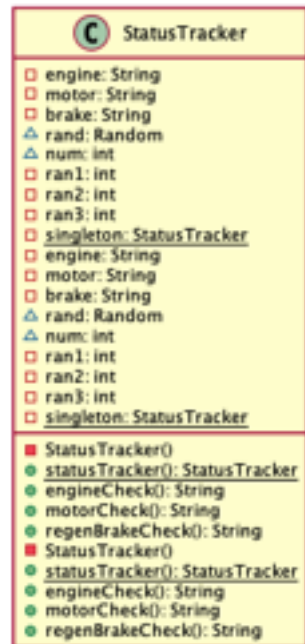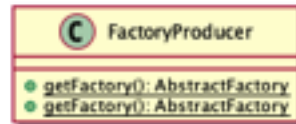
User Manual
-    실행 방법
Javac *.java
Java VisitorDemo.java

Design Document

이 프로그램에는 Visitor, Observer, Singleton, Abstract, Factory, Builder 패턴이 사용되었다. 먼저 프로그램이 시작되면 Motor, Engine, Brake 객체가 생성되고 그에 맞는 observer들을 생성된다. Visitor를 통해 실제로 각 장치들은 검사를 받도록 실행이 된다. 즉 각 visitor가 3개의 장치를 검사하는 방식이다.

Engine
- name: String
- name: String
- Engine(name: String, observer1: Observer)
- getName(): String
- accept(v: Visitor): void
- toString(): String
- notifyObserver(status: String): void
- notifyObserver_sr(status: String): void
- notifyObserver_sw(status: String): void
- getStatus(): String
- Engine(name: String, observer1: Observer)
- getName(): String
- accept(v: Visitor): void
- toString(): String
- notifyObserver(status: String): void
- notifyObserver_sr(status: String): void
- notifyObserver_sw(status: String): void
- getStatus(): String

RegenerativeBrake
- name: String
- name: String
- RegenerativeBrake(name: String, observer3: Observer)
- getName(): String
- accept(v: Visitor): void
- toString(): String
- notifyObserver(status: String): void
- notifyObserver_sr(status: String): void
- notifyObserver_sw(status: String): void
- RegenerativeBrake(name: String, observer3: Observer)
- getName(): String
- accept(v: Visitor): void
- toString(): String
- notifyObserver(status: String): void
- notifyObserver_sr(status: String): void
- notifyObserver_sw(status: String): void

ElectricMotor
- name: String
- name: String
- ElectricMotor(name: String, observer2: Observer)
- getName(): String
- accept(v: Visitor): void
- toString(): String
- notifyObserver(status: String): void
- notifyObserver_sr(status: String): void
- notifyObserver_sw(status: String): void
- ElectricMotor(name: String, observer2: Observer)
- getName(): String
- accept(v: Visitor): void
- toString(): String
- notifyObserver(status: String): void
- notifyObserver_sr(status: String): void
- notifyObserver_sw(status: String): void

observer1 observer1   observer3 observer3   observer2 observer2

Observer
update(status: String): void
getLog(): String
update_sr(status: String): void
update_sw(status: String): void
update(status: String): void
getLog(): String
update_sr(status: String): void
update_sw(status: String): void

EngObserver
- log: String
- log: String
- update(status: String): void
- getLog(): String
- update_sr(status: String): void
- update_sw(status: String): void
- update(status: String): void
- getLog(): String
- update_sr(status: String): void
- update_sw(status: String): void

BrakeObserver
- log: String
- log: String
- update(status: String): void
- getLog(): String
- update_sr(status: String): void
- update_sw(status: String): void
- update(status: String): void
- getLog(): String
- update_sr(status: String): void
- update_sw(status: String): void

MotObserver
- log: String
- log: String
- update(status: String): void
- getLog(): String
- update_sr(status: String): void
- update_sw(status: String): void
- update(status: String): void
- getLog(): String
- update_sr(status: String): void
- update_sw(status: String): void

- System/Component Checking Visitor에서는 3개의 장치를 방문한다. 이때 singleton에 의해 생성된 각 부분의 status를 바탕으로 상태를 점검한다. Singleton은 Random한 숫자로 각 상태를 결정한 후 이를 3개의 Visitor에서 모두 사용한다. Singleton을 통해 알게된 state들을 바탕으로 visitor는 각 장치를 처리한다. Singleton UML은 다음과 같다.

- 각각의 visitor는 총 3개의 Visit()으로 구성되어 있는데 이는 각각은 3개의 장치에 따라 다르게 호출된다. 그래서 각각 필요한 처리 과정은 visitor들이 호출될 때 어떤 장치에서 호출하는지에 따라 다르게 구성하였다. 밑에 UML은 3개의 visitor의 구조를 나타낸다.
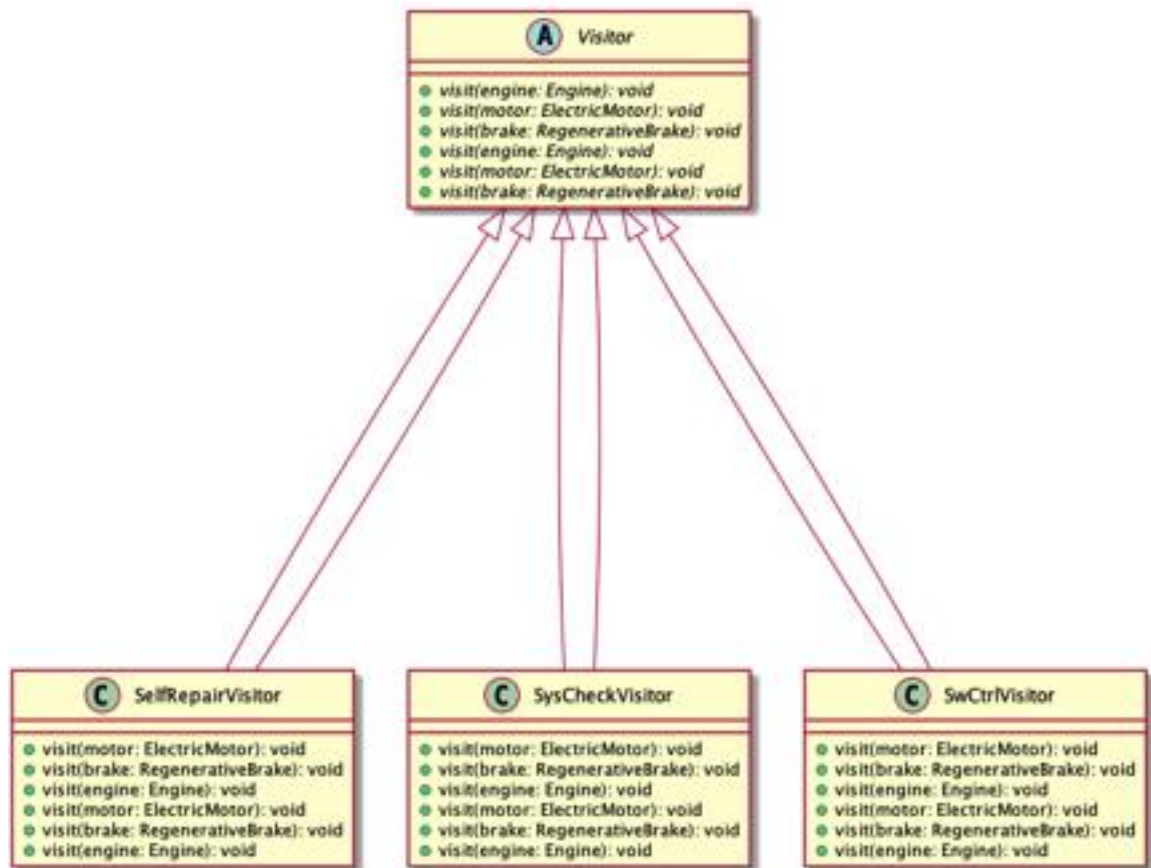
- Self Repairing visitor와 software Control visitor 역시 3개의 visit()으로 되어 있고 각각은 다른 장치를 검사한다.

- Engine, Motor, Brake는 각각 자신의 상태를 Observer에게 알린다. 그래서 현재의 log를 observer들이 updatae하도록 한다. 또한 System/Component Checking Visitor에 의한 observer 결과는 후에 builder에게 알려줘 html 파일을 생성하게 된다. 이때 Builder Pattern이 사용된다. Director class에서 html에 대한 정보를 가지고 파일을 build 하면, Builder class에 의해 정의된 HtmlBuilder가 형식에 맞춰서 director가 준 정보를 바탕으로 실제로 파일을 생성하게 된다. 밑에 UML은 Builder pattern의 구조를 나타낸다.

Director

- Director(builder: Builder)
- construct(observer1: Observer, observer2: Observer, observer3: Observer): void
- Director(builder: Builder)
- construct(observer1: Observer, observer2: Observer, observer3: Observer): void

Message

printer(): void
printer(): void

builder   builder

1   1

Builder

- makeTitle(title: String): void
- makeString(str: String): void
- makeItems(items: String[]): void
- close(): void
- makeTitle(title: String): void
- makeString(str: String): void
- makeItems(items: String[]): void
- close(): void

Print_HW

- printer(): void
- printer(): void

print_SW1

- printer(): void
- printer(): void

print_SW

- printer(): void
- printer(): void

HtmlBuilder

- filename: String
- writer: PrintWriter
- filename: String
- writer: PrintWriter

- makeTitle(title: String): void
- makeString(str: String): void
- makeItems(items: String[]): void
- close(): void
- getResult(): String
- makeTitle(title: String): void
- makeString(str: String): void
- makeItems(items: String[]): void
- close(): void
- getResult(): String

- Html의 실제 결과는 다음과 같다. 첫번째 Visitor의 결과를 나타내도록 하였다.

log

Gasoline Engine Being Checked

- Checking Engine
- Engine is OK

Checked Eletric Motor

- Motor Checked
- Electric Motor is vibratingMotor

Checking Regenerative Brake System

- Checking Regenerative Brake System
- Generative Brake System is abnormalSpeedDecrease

- 위에 UML에는 Message 클래스가 있는데 이는 Factory Pattern이다. 특정한 조건이 있을 경우, Factory에서 조건에 맞도록 정해진 문장을 출력한다. 이때에는 Abstract Factory Pattern이 사용되어, abstract class로 Factory의 구조를 정의한다. 해당 시스템에서는 3가지의 특정 조건에서는 Factory pattern이 필요하였기 때문에, HW와 2개의 SW로 각각 조건에 맞는 출력을 하도록 하는 class들이 정의되도록 프로그램을 구성하였다. 밑에는 이를 표현한 UML이다.

```
┌─────────────────────────────────────────┐
│  Ⓒ  AbstractFactory                      │
├─────────────────────────────────────────┤
│ ▲ getMessage(type: String): Message      │
│ ▲ getMessage(type: String): Message      │
└─────────────────────────────────────────┘
                  △ △
                  │ │
                  │ │
                  │ │
                  │ │
                  │ │
┌─────────────────────────────────────────┐
│  Ⓒ  Factory                              │
├─────────────────────────────────────────┤
│ ● getMessage(type: String): Message      │
│ ● getMessage(type: String): Message      │
└─────────────────────────────────────────┘
```