CS350 Introduction to Software Engineering, Spring 08

SafeHome Team 4

Design Model
Version 1.0
Printed by Team4

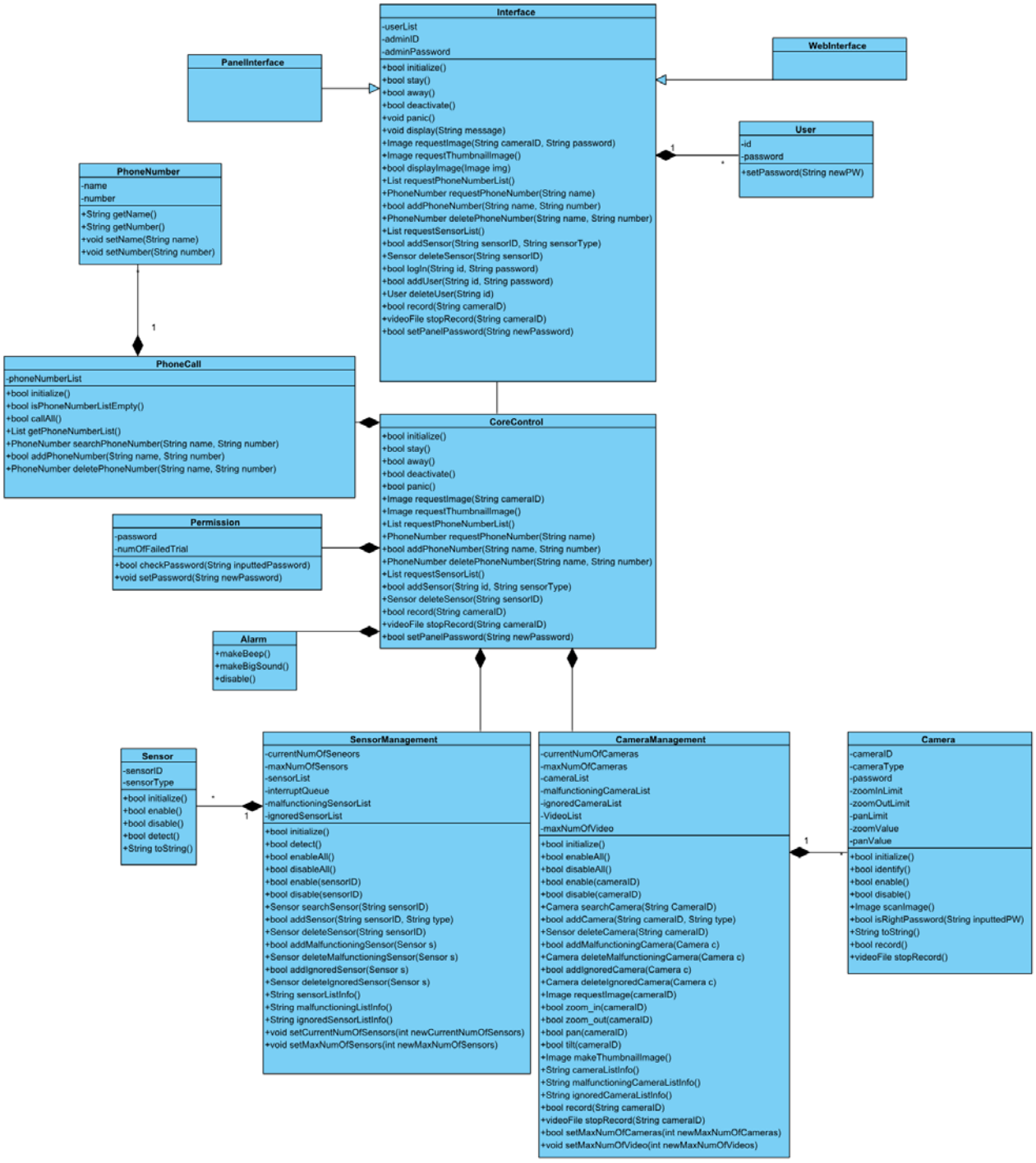
Team 4 :
20000330 Sangyoung, Lee
20040033 Sejoong, Kwon
20050426 Jieun, Lee
20070485 Kanghee, Won

Table of Contents

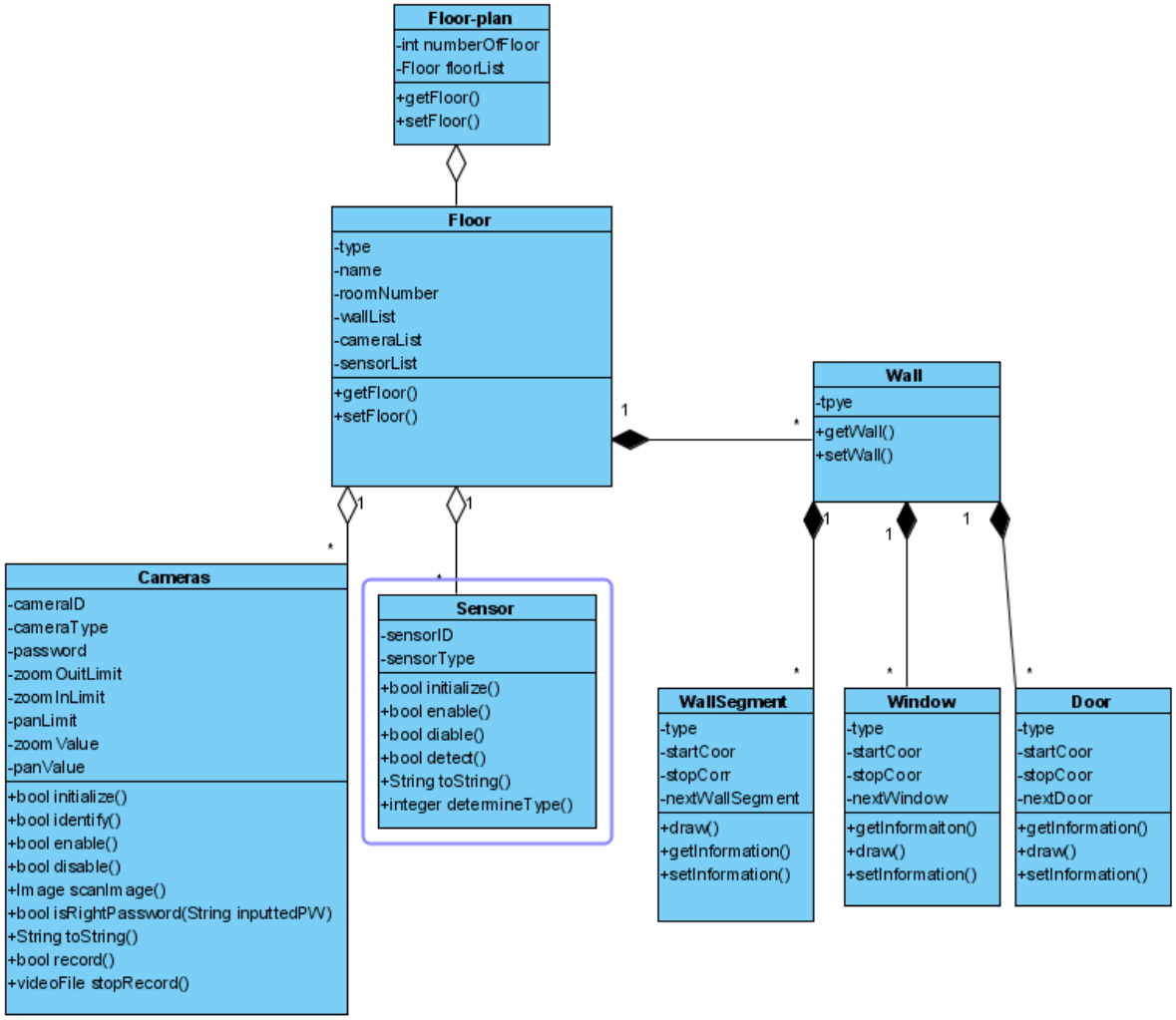
1. Class Diagram

1.1 Class Diagram

1.1.1 Main class diagram

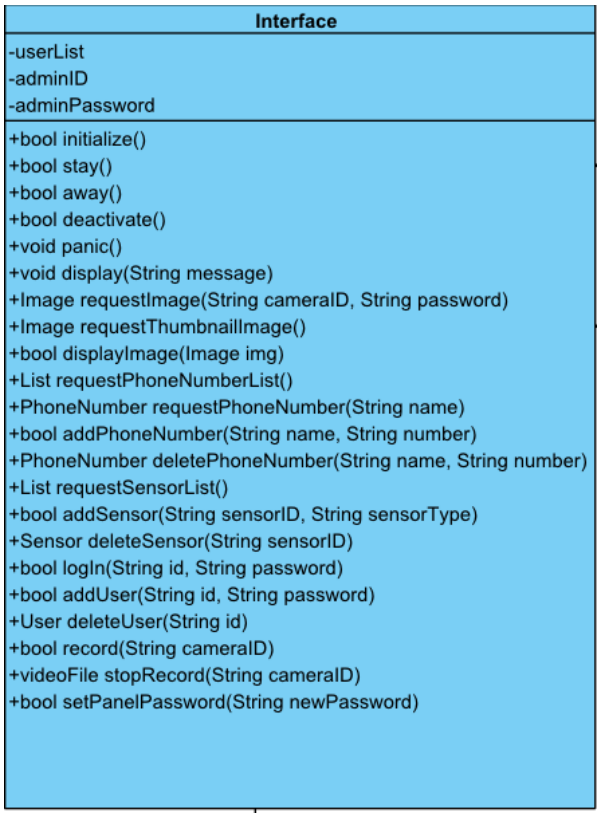


1.1.2 Floor-plan



1.2. Class Description

1.2.1. Interface



1. Description

This class is for interface which is panel and internet in this Safehome system. This class has most of function that must be implemented. This class is super class of PanelInterface and WebInterface. The PanelInterface andWebInterface inherits all attributes and operations in the Interface.

Panel and internet class have all the functions. And when some input from the external device is occurred and they call this function, this class command “CoreControl” class to do something. The CoreContol class is the core of system. So, this interface class that order to CoreControl class is the most superior class.

2. Attribute

1. **userList:** This list concludes all users who accesses via internet. This userList composed of User Objects which have id and password as its attributes. This userList has the list of User Objects. And administrator is not included in the userList.
2. **adminID:** SafeHome only have on user as a Administrator. And this attribute means the administrator’s ID
3. **adminPassword:** Administrator’s password.

3. Responsibilities

1. **bool initialize()**
   - Precondition: SafeHome HW box, panel must be installed.

- Postcondition: SafaHome system is initialized.
- Trigger: When user turns on the power of safeHome system.
- This function used when safeHome power turns on or off. If panel interface or web interface request initialize function, initialize function in the coreControl is conducted to initialize all classes. If this function is properly done, true is returned.

2. **bool stay()**
   - Precondition: SafeHome is deactivated state.
   - Postcondition:SafeHome is "stay" activated.
   - Trigger: When user using panel pushes "stay" button on the panel.
     When user accessing via web interface "stay" activates the safeHome.
   - This function used when user "stay" activates safeHome system. If this function is properly done, true is returned.

3. **bool away()**
   - Precondition: SafeHome is deactivated state.
   - Postcondition: SafaHome is "away" activated.
   - Trigger: When user using panel pushes "away" button on the panel.
     When user accessing via web interface "away" activates the safeHome.
   - This function used when user "stay" activates safeHome system. If this function is properly done, true is returned.

4. **bool deactivate()**
   - Precondition: SafeHome is activated state.
   - Postcondition: SafaHome is deactivated
   - Trigger: When user using panel inputs right password on the panel.
     When user accessing via web interface deactivates the safeHome.
   - This function used when user deactivates safeHome system. If this function is properly done, true is returned.

5. **void panic()**
   - Precondition: User using panel pushes panic button.
   - Postcondition: SafeHome syste alarms and calls the number in the phoneCall list.
   - Trigger: When user using panel pushes panic button.
   - This function is used when user pushes panic button. If this button is pushed, coreControl class's panic() function is called. And in the previous functions, alarm class's makeBigSound() and PhoneCall class's callAll() function is called.

6. **void display(String message)**
   - Precondition: There is message to be displayed in the panel display.
   - Postcondition: Panel displays messages.
   - Trigger: There is message to be displayed in the panel display.
   - While doing the initialize() functions, if there are some malfunctioning sensors or malfunctioning cameras, panel must displays the information of malfunctioning sensors cameras. So display(String message) function is needed.

7. **Image requestImage(String id, String password)**
   - Precondition: User accessing via internet requests a camera view, and inputs password of the camera.

- Postcondition: Selected camera's view is returned.
- Trigger: When user accesing via web interface requests a camera view.
- This function calls the CoreControl's requestImage(String id, String password), and this function calls to CameraMangement's requestImage(String id, String password) and the camera which has that id return images.

8. **Image requestThumbnailImage()**
   - Precondition: User accessing via internet requests a thumbnail view
   - Postcondition: Thumbnail view is returned.
   - Trigger: When user accesing via web interface requests a thumbnail view.
   - This function calls the CoreControl's requestThumbnailImage(), and this function calls to CameraMangement's makeThumbnailImage(). And then maked image is returned.

9. **bool displayImage(Image img)**
   - Precondition: User accessing via internet requests any kind of camera view.
   - Postcondition: Web interface displays the image.
   - Trigger: When User accessing via internet requests any kind of camera view.
   - This function displays the img to web interface.

10. **List requestPhoneNumberList()**
    - Precondition: User accessing via internet requests PhoneNumber list
    - Postcondition: Web interface displays the PhoneNumber list.
    - Trigger: When User accessing via internet requests PhoneNumber list.
    - This function displays the PhoneNumber list.

11. **PhoneNumber requestPhoneNumber(String name)**
    - Precondition: User accessing via internet requests PhoneNumber list
    - Postcondition: Web interface displays the PhoneNumber list.
    - Trigger: When User accessing via internet requests PhoneNumber list.
    - This function displays the PhoneNumber list.

12. **bool addPhoneNumber(String name, String number)**
    - Precondition: User accessing via internet requests to add phoneNumber to PhoneNumber list
    - Postcondition: A new phoneNumber is added to PhoneNumber list.
    - Trigger: When User accessing via internet adds phoneNumber to PhoneNumber list.
    - This function adds new phoneNumber to the PhoneNumber list.

13. **PhoneNumber deletePhoneNumber(String name, String number)**
    - Precondition: User accessing via internet requests to delete phoneNumber from PhoneNumber list
    - Postcondition: A selected phoneNumber is deleted to PhoneNumber list.
    - Trigger: When User accessing via internet request to delete phoneNumber from PhoneNumber list.
    - This function deletes the selected phoneNumber from the PhoneNumber list.

14. **List requestSensorList()**
    - Precondition: User accessing via internet requests Sensor list

- Postcondition: SensorList is returned.
- Trigger: When User accessing via internet requests Sensor list.
- This function returns sensorList.

15. **bool addSensor(String sensorID, String sensorType)**
    - Precondition: User accessing via internet requests to add a new Sensor to SensorList.
    - Postcondition: The new sensor is added to SensorList.
    - Trigger: When User accessing via internet requests to add a new Sensor.
    - This function adds new Sensor to SensorList.

16. **Sensor deleteSensor(String sensorID)**
    - Precondition: User accessing via internet requests to delete a selected Sensor from SensorList.
    - Postcondition: The selected sensor is deleted from the SensorList.
    - Trigger: When User accessing via internet requests to delete a selected Sensor.
    - This function deletes a selected Sensor to SensorList.

17. **bool logIn(String id, String password)**
    - Precondition: User requests to log in the web interface.
    - Postcondition: User log-in is accepted.
    - Trigger: When User requests to log in the web interface.
    - This function accepts user who properly logs in the web interface by inputting right id and password. If the log-in procedure is properly done, this function returns true.

18. **bool addUser(String id, String password)**
    - Precondition: User requests to add a new user to web interface.
    - Postcondition: The new user is added.
    - Trigger: When User requests to add a new user to web interface.
    - This function adds a new user. If the adding procedure is properly done, this function returns true.

19. **User deleteUser(String id)**
    - Precondition: User requests to delete a selected user from web interface.
    - Postcondition: The selected user is deleted
    - Trigger: When User requests to delete a selected user from web interface.
    - This function deletes a selected user. The deleted user object is returned.

20. **bool record(String cameraID)**
    - Precondition: User requests to start recording a view of the selected camera.
    - Postcondition: The camera recording is started.
    - Trigger: When User requests to start recording a view of the selected camera.
    - This function records view of the selected camera. If the starting to record is properly done, this function returns true.

21. **videoFile stopRecord(String cameraID)**
    - Precondition: User requests to end recording a view of the selected camera.
    - Postcondition: The camera recording is ended.
    - Trigger: When User requests to end recording a view of the selected camera.

- This function records view of the selected camera. If the stopping to record is properly done, this function returns the video which is recorded by previous procedure.

22. **setPanelPassword(String newPassword)**
    - Precondition: User requests to modify panel password.
    - Postcondition: The panel password is modified.
    - Trigger: When User requests to modify panel password.
    -This function modifies panel password. This function returns true if panel password properly changed.

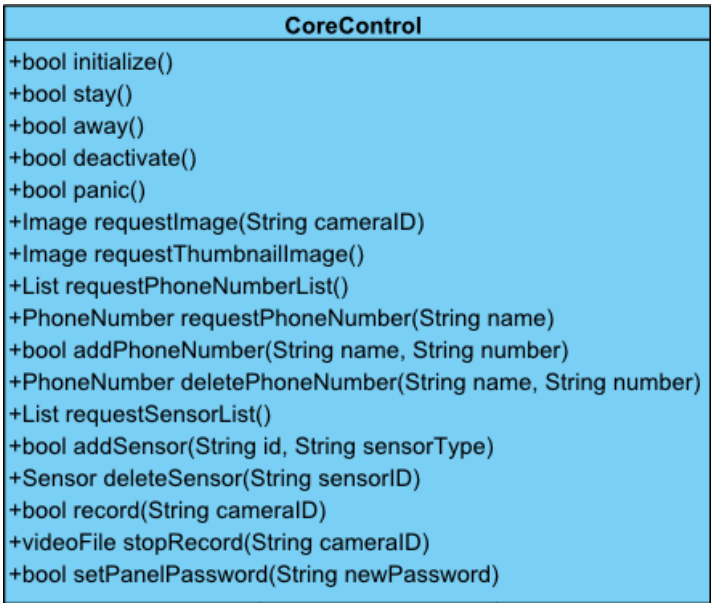
4. Traceability

Responsibilities	**Use Case Description**
+bool initialize()	3.1 Initialization
+bool stay()	3.2 Activation
+bool away()	3.2 Activation
+bool deactivate()	3.3 Deactivation
+void panic()	3.7 Alert
+void display(String message)	3.1 Initialization, 3.2 Activation
+Image requestImage(String CameraID, String password)	3.6 Access Cameras
+Image requestThumbnailImage()	3.6 Access Cameras
+ListrequestPhoneNumberList()	3.8 Configuration
+PhoneNumber requestPhoneNumber(String name)	
+bool addPhoneNumber(String name,String number)	3.8 Configuration
+PhoneNumber deletePhoneNumber(String name,String number)	3.8 Configuration
+List requestSensorList()	3.5 Access Sensors
+bool addSensor(String sensorID, String sensorType)	3.8 Configuration
+Sensor deleteSensor(String sensorID)	3.8 Configuration
+bool login(String id, String password)	Newly added
+bool addUser(String id, String password)	Newly added
+User searchUser(String id)	Newly added
+void setPanelPassword(String newPassword)	3.8 Configuration

5. CRC Card

Interface	
Super Classes:	
Sub Classes: PanelInterface, WebInterface	
Description: Interface class represents input device of safeHome systems like panel, and web.	
Attributes:	
Name	Description
userList	User list which composed of users who are accessing via internet
adminID	administrator's ID
adminPassword	administrator's password
Responsibilities:	
Name	Collaborator
initializes all modules	CoreControl
stay activates the safeHome system	CoreControl
away activates safeHome system	CoreControl
deactivates safeHome system	CoreControl
presses panic button	CoreControl
displays messages to panel	CoreControl
requests camera's image on web interface	CoreControl
requests Thumbnail image on web interface	CoreControl
displays image on web	CoreControl
requests PhoneNumberList on web interface	CoreControl
requests PhoneNumber	CoreControl
adds PhoneNumber with name, and number	CoreControl
deletes PhoneNumber with name and number	CoreControl
requests SensorList	CoreControl
adds Sensor with sensor type	CoreControl
deletes Sensor with sensor id	CoreControl
logs in the web Interface	CoreControl
adds user on the web interface with name and password	CoreControl
deletes user on the web with id	CoreControl
records camera's view with cameraID	CoreControl
stops recording camera's view with cameraID	CoreControl
sets panel password with new password	CoreControl
sets maximum number of sensors with new number	CoreControl
enables camera whose id is cameraID	CoreControl
disables camera whose id is cameraID	CoreControl
enables sensor whose id is sensorID	CoreControl
disables sensor whose ie is sensorID	CoreControl

1.2.2. CoreControl



1. Description

This is CoreControl class. This class represents the core of system. This class is the superior class than other classes without Interface classes. Other functionality classes are CameraManagement, SensorManagement, Permission, Alarm, PhoneCall and so on, connected from this class. And this CoreControl class achieve results of the function by controlling other class.

2. Attribute

None

3. Responsibilities

1. **bool initialize()**
   - Precondition: SafeHome HW box, panel must be installed.
   - Postcondition: SafaHome system is initialized.
   - Trigger: When user turns on the power of safeHome system.
   - This function used when safeHome power turns on or off. If panel interface or web interface request initialize function, this function is called, and this function initializes other classes. If this function is properly done, true is returned.

2. **bool stay()**
   - Precondition: SafeHome is deactivated state.
   - Postcondition: SafeHome is “stay” activated.
   - Trigger: When user using panel pushes “stay” button on the panel.
     When user accessing via web interface “stay” activates the safeHome.
   - This function used when user “stay” activates safeHome system. The interface class calls this function to activate the safeHome system. If this function is properly done, true is returned.

3. **bool away()**
   - Precondition: SafeHome is deactivated state.
   - Postcondition: SafaHome is “away” activated.

- Trigger: When user using panel pushes “away” button on the panel.
When user accessing via web interface “away” activates the safeHome.
- This function used when user “stay” activates safeHome system. The interface class calls this function to activate the safeHome system. If this function is properly done, true is returned.

4. **bool deactivate()**
- Precondition: SafeHome is activated state.
- Postcondition: SafaHome is deactivated
- Trigger: When user using panel inputs right password on the panel.
When user accessing via web interface deactivates the safeHome.
- This function used when user deactivates safeHome system. The interface class calls this function to deactivate the safeHome system.

5. **void panic()**
- Precondition: User using panel pushes panic button.
- Postcondition: SafeHome syste alarms and calls the number in the phoneCall list.
- Trigger: When user using panel pushes panic button.
- This function is used when user pushes panic button. If this button is pushed, Interface class’s panic() function calls this function. And this function calls alarm class’s makeBigSound() and PhoneCall class’s callAll() function.

6. **Image requestImage(String id, String password)**
- Precondition: User accessing via internet requests a camera view, and inputs password of the camera to web interface.
- Postcondition: Selected camera’s view is returned.
- Trigger: When user accesing via web interface requests a camera view.
- This function is called when Interface’s requestImage(String id, String password) is called. And this function calls CameraMangement’s requestImage(String id, String password) and the camera which has that id return images.

7. **Image requestThumbnailImage()**
- Precondition: User accessing via internet requests a thumbnail view
- Postcondition: Thumbnail view is returned.
- Trigger: When user accesing via web interface requests a thumbnail view.
- This function is called when Interface’s requestThumbnailImage() is called. And this function calls to CameraMangement’s makeThumbnailImage(). And then maked image is returned.

8. **List requestPhoneNumberList()**
- Precondition: User accessing via internet requests PhoneNumber list
- Postcondition: Web interface displays the PhoneNumber list.
- Trigger: When User accessing via internet requests PhoneNumber list.
- This function returns the PhoneNumber list.

9. **PhoneNumber requestPhoneNumber(String name)**
- Precondition: User accessing via internet requests PhoneNumber list
- Postcondition: Web interface displays the PhoneNumber list.
- Trigger: When User accessing via internet requests PhoneNumber list.

- This function returns the PhoneNumber object which has the name in the phone Number List.

10. **bool addPhoneNumber(String name, String number)**
    - Precondition: User accessing via internet requests to add phoneNumber to PhoneNumber list
    - Postcondition: A new phoneNumber is added to PhoneNumber list.
    - Trigger: When User accessing via internet adds phoneNumber to PhoneNumber list.
    - This function is called when Interfaces addPhoneNumberList(String name, String number)is called. This function return true when adding procedure is properly done.

11. **PhoneNumber deletePhoneNumber(String name, String number)**
    - Precondition: User accessing via internet requests to delete phoneNumber from PhoneNumber list
    - Postcondition: A selected phoneNumber is deleted to PhoneNumber list.
    - Trigger: When User accessing via internet request to delete phoneNumber from PhoneNumber list.
    - This function is called when Interfaces deletePhoneNumberList(String name, String number)is called. This function returns the deleted PhoneNumber object when deleting procedure is properly done.

12. **List requestSensorList()**
    - Precondition: User accessing via internet requests Sensor list
    - Postcondition: SensorList is returned.
    - Trigger: When User accessing via internet requests Sensor list.
    - This function is called when Interface requestSensorList() function is called. And this function returns sensorList.

13. **bool addSensor(String sensorID, String sensorType)**
    - Precondition: User accessing via internet requests to add a new Sensor to SensorList.
    - Postcondition: The new sensor is added to SensorList.
    - Trigger: When User accessing via internet requests to add a new Sensor.
    - This function adds new Sensor to SensorList.

14. **Sensor deleteSensor(String sensorID)**
    - Precondition: User accessing via internet requests to delete a selected Sensor from SensorList.
    - Postcondition: The selected sensor is deleted from the SensorList.
    - Trigger: When User accessing via internet requests to delete a selected Sensor.
    - This function is called when interface's deleteSensor(String sensorID) is called. This function returns deleted Sensor object.

15. **bool record(String cameraID)**
    - Precondition: User requests to start recording a view of the selected camera.
    - Postcondition: The camera recording is started.
    - Trigger: When User requests to start recording a view of the selected camera.
    - This function records view of the selected camera. If the starting to record is

properly done, this function returns true.

16. **videoFile stopRecord(String cameraID)**
    - Precondition: User requests to end recording a view of the selected camera.
    - Postcondition: The camera recording is ended.
    - Trigger: When User requests to end recording a view of the selected camera.
    - This function records view of the selected camera. If the stopping to record is properly done, this function returns the video which is recorded by previous procedure.

17. **bool setPanelPassword(String newPassword)**
    - Precondition: User requests to modify panel password.
    - Postcondition: The panel password is modified.
    - Trigger: When User requests to modify panel password.
    -This function modifies panel password. This function is called when Interface's setPanelPassword(String newPassword) is called. This function returns true if panel password properly changed.

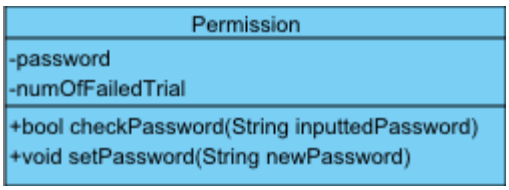
4. Traceability

Responsibilities	**Use Case Description**
+bool initialize()	3.1 Initialization
+bool stay()	3.2 Activation
+bool away()	3.2 Activation
+bool deactivate()	3.3 Deactivation
+void panic()	3.7 Alert
+Image requestImage(String CameraID, String password)	3.6 Access Cameras
+Image requestThumbnailImage()	3.6 Access Cameras
+ListrequestPhoneNumberList()	3.8 Configuration
+PhoneNumber requestPhoneNumber(String name)	Newly added
+bool addPhoneNumber(String name,String number)	3.8 Configuration
+PhoneNumber deletePhoneNumber(String name,String number)	3.8 Configuration
+List requestSensorList()	3.5 Access Sensors
+bool addSensor(String sensorID, String sensorType)	Newly added
+Sensor deleteSensor(String sensorID)	3.8 Configuration
+bool record(String cameraID)	3.6 Access Cameras
+ImgFile stopRecord(String CameraID)	3.6 Access Cameras
+void setPanelPassword(String newPassword)	3.8 Configuration

5. CRC Card

CoreControl	
Super Classes:	
Sub Classes:	
Description: Panel core control class.	
Attributes:	
Name	Description
Responsibilities:	
Name	Collaborator
initializes all modules	
stay activates the safeHome system	
away activates the safeHome system	
deactivates the safeHome system	
presses panic button	
requests camera image	CameraManagement
requests thumbnail image	CameraManagement
requests PhoneNumberList	PhoneCall
requests PhoneNumber with name and numbe	PhoneCall
adds PhoneNumber with name and number	PhoneCall
deletes PhoneNumber with name and number	PhoneCall
requests SensorList	SensorManagement
adds sensor with id and type	SensorManagement
deletes sensor with id	SensorManagement
records camera's view with cameraID	CameraManagement
stops recording camera's viw with cameraID	CameraManagement
sets panel password with new password	Permission
sets maximum number of sensors	SensorManagement
enables camera whose id is cameraID	CameraManagement
disables camera whose id is cameraID	CameraManagement
enables sensor whose id is sensorID	SensorManagement
disables sensor whose id is sensorID	SensorManagement

1.2.3 Permission



1. Description

When user inputs 4-digit password to panel to deactivate the safeHome system, the inputted password must be checked. Or, administrator can change panel password via web interface. So this class deals those responsibilities.

2. Attribute

1. password: panel password.
2. numOfFailedTrial: number of failed trial.

3. Responsibilities

1. **bool checkPassword(String inputtedPassword)**
   - Precondition: User inputted password.
   - Postcondition: User gets the permission.
   - Trigger: User inputted password, and Core Controller wants to confirm that the inputted password is correct or not.
   - This function is called when the user inputted the password into panel or internet to get the permission.

2. **bool setPassword(String newPassword)**
   - Precondition: Administrator via internet wants to modify the password.
   - Postcondition: The password is changed.
   - Trigger: Administrator via internet orders modifying password.
   - In CoreControl class, there is setPassword(String newPassword) functions. In that function this function is called and set newPassword as a password

4 .Traceability

Responsibilities	**Use case description**
+bool checkPassword(String inputtedPassword)	3.4 Permission
+void setNewPassword(String newPassword)	3.8 Configuration

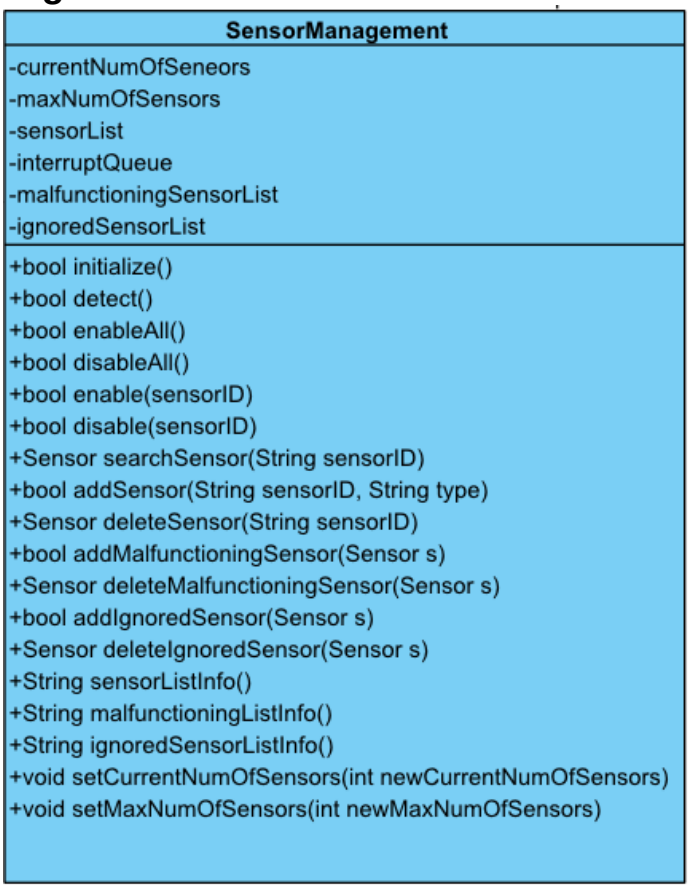
5. CRC Card

Permission	
Super Classes:	
Sub Classes:	
Description:	Permission class checks the password which inputted by use is correct or not.

Attributes:	
Name	Description
password	current correct password(4-digit)
numOfFailedTrial	number of failed trials

Responsibilities:	
Name	Collaborator
checks that password which is inputted by user is correct	
sets new password via web interface	

1.2.4 SensorManagement



1. Description

This class is for sensor management. All functions related to sensor management are accomplished in this class. User accessing internet can turn on or off certain sensors, and add or delete sensors. And also user can modify maximum number of sensors.

2. Attribute

1. **maxNumOfSensors**: maximum number of sensors that can exist in the house.
2. **numOfEnabledSensors**: current number of enabled sensors.
3. **sensorList**: all sensor list in the house.
4. **enabledSensorList**: enabled sensor list.
5. **malfunctioningSensorList**: list of malfunctioning sensors.
6. **ignoredSensorList**: list of ignored sensors.
7. **interruptQueue**: queue of sensor interrupts.

3. Responsibilities

1. **bool initialize()**

- Precondition: CoreContol's initialize() function is called.
- Postcondition: Sensors are initialized.
- Trigger: When user turns on the power of safeHome system.
  When panel is "away" activated.
- This function is used when safeHome power turns on. And whenever the system is

"away" activated, SensorManagement's initialize() function must be called. If this function is properly done, and sensors are properly initialized, true is returned.

2. **bool detect()**
   - Precondition: SafeHome is "away" activated.
   - Postcondition: All sensors wait for interrupts.
   - Trigger: When SafeHome is "away" activated.
   - This function is used when safeHome is "away" activated. And whenever the system is "away" activated, SensorManagement's detect() function must be called to detect motions in the house. If this function is properly done, true is returned.

3. **bool enableAll()**
   - Precondition: SafeHome is "away" activated.
   - Postcondition: All sensors are enabled.
   - Trigger: When SafeHome is "away" activated.
   - This function is used when safeHome is "away" activated. And whenever the system is "away" activated, SensorManagement's enables all sensors in the house. So this function calls all sensors' enable() function. If this function is properly done, true is returned.

4. **bool disableAll()**
   - Precondition: SafeHome is deactivated.
   - Postcondition: All sensors are disabled.
   - Trigger: When SafeHome is deactivated.
   - This function is used when safeHome is deactivated. And whenever the system is deactivated, SensorManagement's disables all sensors in the house. So this function calls all sensors' disable() function. If this function is properly done, true is returned.

5. **bool enable(sensorID)**
   - Precondition: User accessing via web interface enables sensor whose id is sensorID.
   - Postcondition: sensor which has sensorID is enabled.
   - Trigger: When user accessing via web interface enables sensor whose id is sensorID.
   - This function is used when safeHome is "away" activated and user accessing via web interface enables certain sensor whose id is sensorID. So this function calls the selected sensor's enable() function. If this function is properly done, true is returned.

6. **bool disable(sensorID)**
   - Precondition: User accessing via web interface disables sensor whose id is sensorID.
   - Postcondition: sensor which has sensorID is disabled.
   - Trigger: When user accessing via web interface disables sensor whose id is sensorID.
   - This function is used when safeHome is "away" activated and user accessing via web interface disables certain sensor whose id is sensorID. So this function calls the selected sensor's disable() function. If this function is properly done, true is returned.

7. **Sensor searchSensor(String sensorID)**
   - Precondition: None
   - Postcondition: Sensor which has sensorID is returned.

- Trigger: When functions which needs to search certain sensor with sensorID calls this function.
- This function is used when functions which need to search certain sensor with sensorID are called. If this function is properly done, sensor whose id is sensorID is returned.

8. **bool addSensor(String sensorID, String type)**
   - Precondition: User accessing vie web interface requests to add new sensor which has sensorID and type to sensorList.
   - Postcondition: A new sensor is added to sensorList.
   - Trigger: When user accessing vie web interface requests to add new sensor which has sensorID and type.
   - This function used when user accessing via web interface requests to add new sensor. If this function is properly done, sensor whose id is sensorID and type is type is added to sensorList. A newly added sensor is initially diabled. And if this function is properly done, sensor whose id is sensorID and type is type is added.

9. **Sensor deleteSensor(String sensorID)**
   - Precondition: User accessing vie web interface requests to delete sensor which has sensorID from sensorList.
   - Postcondition: The selected sensor is deleted form sensorList.
   - Trigger: When user accessing vie web interface requests to delete sensor which has sensorID from sensorList.
   - This function used when user accessing via web interface requests to delete sensor whose id is sensorID. If this function is properly done, sensor is deleted from sensorList. And if this function is properly done, sensor whose id is sensorID returned.

10. **bool addMalfunctioningSensor(Sensor s)**
    - Precondition: None.
    - Postcondition: A malfunctioning sensor is added to malfunctioningSensorList.
    - Trigger: While accomplishing any functions related to sensors, if one sensor is malfunctioning, this function is called.
    - If malfunctioning sensor is found, the sensor is added to malfunctioningSensorList. And if this function is properly done, true is returned.

11. **Sensor deleteMalfunctioningSensor(Sensor s)**
    - Precondition: None.
    - Postcondition: Sensor s is deleted from malfunctioningSensorList.
    - Trigger: When malfunctioning sensor is fixed or trashed.
    - This function deletes Sensor s from the malfunctioningSensorList. And if this function is properly done, the deleted Sensor is returned.

    Sensor deleteIgnoredSensor(Sensor s)
    - Precondition: None
    - Postcondition: Sensor s is deleted from ignoredSensorList.
    - Trigger: indecisive
    - This function deletes Sensor s from the ignoredSensorList. And if this function is properly done, the deleted Sensor is returned.

12. **String sensorListInfo()**
    - Precondition: When user accessing via web interface requests sensorList information.
    - Postcondition: sensorList information is made and returned.
    - Trigger: When user accessing via web interface requests sensorList information.
    - This function makes String which contains information about sensors which is in sensorList. And if this function is properly done, the String which made previously is returned.
13. **String malfunctioningListInfo()**
    - Precondition: None
    - Postcondition: malfunctioningSensorList information is made and returned.
    - Trigger: indecisive
    - This function makes String which contains information about sensors which is in malfunctioningSensorList. And if this function is properly done, the String which made previously is returned.

14. **String ignoredSensorListInfo()**
    - Precondition: None
    - Postcondition: malfunctioningSensorList information is made and returned.
    - Trigger: indecisive
    - This function makes String which contains information about sensors which is in ignoredSensorList. And if this function is properly done, the String which made previously is returned.

15. **void setNumOfEnabledSensors(int newCurrentNumOfSensors)**
    - Precondition: User enables or disables sensors.
    - Postcondition: The current number of enabled sensors is modified.
    - Trigger: When current number of enabled sensor is changed.
    -When user accessing via web interface enables or disables sensors the numOfEnabledSensors should be changed. This function modifies numOfEnabledSensors. This function returns true if numOfEnabledSensors is properly changed.

16. **void setMaxNumOfSensors(int newMaxNumOfSensors)**
    - Precondition: User requests to modify the maximum number of sensors in the house.
    - Postcondition: The maximum number of sensors is modified.
    - Trigger: When User requests to modify the maximum number of sensors.
    -This function modifies maximum number of sensors. This function returns true if maximum number of sensors is properly changed.

4. Traceability

Responsibilities	**Use Case Description**
+bool initialize()	3.1 Initialization
+bool detect()	3.7 Alert
+bool enableAll()	3.2 Activation, 3.5 Access Sensors
+bool disableAll()	3.2 Activation, 3.5 Access Sensors
+bool enable(sensorID)	3.5 Access Sensors
+bool disable(sensorID)	3.5 Access Sensors

+Sensor searchSensor(String sensorID, String type)	Newly added
+bool addSensor(String sensorID, String type)	3.8 Configuration
+Sensor deleteSensor(String sensorID)	3.8 Configuration
+bool addMalfunctioningSensor(Sensor s)	3.1 Initialization, 3.2 Activation
+Sensor deleteMalfunctioningSensor(Sensor s)	3.1 Initialization, 3.2 Activation
+bool addIgnoredSensor(Sensor s)	3.2 Activation
+bool deleteIgnoredSensor(Sensor s)	3.2 Activation
+String sensorListInfo()	Newly added
+String malfunctioningListInfo()	Newly added
+String ignoredSensorListInfo()	Newly added
+void setNumOfEnabledSensors(int newCurrentNumOfSensors)	3.8 Configuration
+void setMaxNumOfSensors(int newMaxNumOfSensors)	3.8 Configuration

5. CRC Card

SensorManagement

Super Classes:

Sub Classes:

Description: Activate or deactivate function needs sensor informations when they are functioning. And user can mod

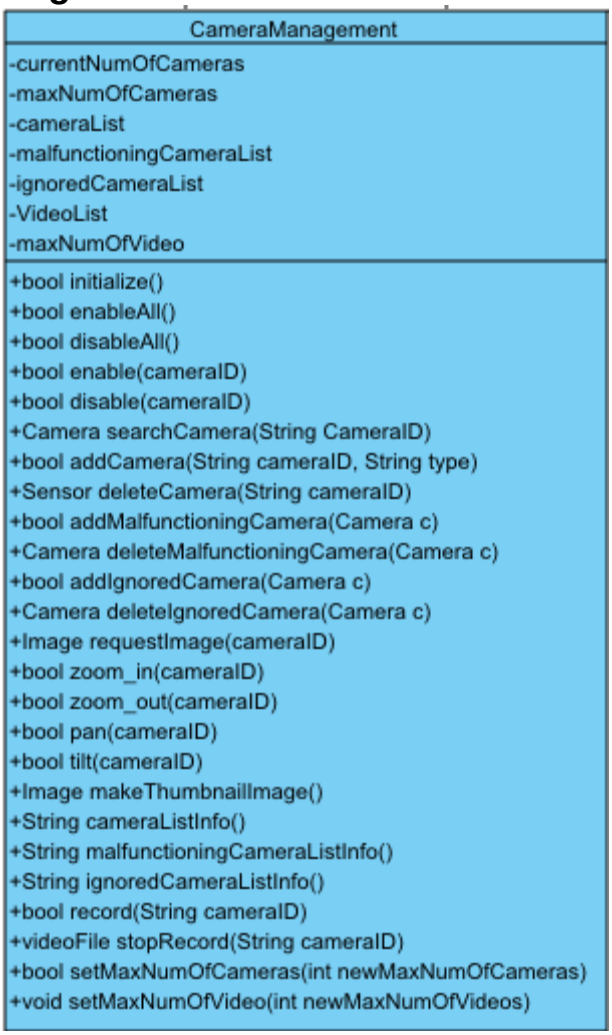
Attributes:

Name	Description
numOfEnabledSensors	current number of sensors
maxNumOfSensors	maximum number of sensors
sensorList	list of sensors which is currently enabled
enabledSensorList	list of enabled sensor list.
malfunctioningSensorLIst	list of malfunctioning sensor list
ignoredSensorList	list of ignored sensor list
interruptQueue	queue of sensor interrupt(detection)

Responsibilities:

Name	Collaborator
initializes all sensors	Sensor
detects sensor interrupt	
enables all sensors at once	Sensor
disables all sensors at once	Sensor
enables sensor which has the sensorID	Sensor
diables sensor which has the sensorID	Sensor
searches sensor with sensorID	
adds sensor with sensorID and type	
deletes sensor with sensorID	
adds malfunctioning sensor with the sensor object	
deletes malfunctioning sensor with the sensor object	
adds ignored sensor with the sensor object	
deletes ignored sensor with the sensor object	
makes string about sensorList	Sensor
makes string about malfunctioningSensorList	Sensor
makes string about ignoredSensorList	Sensor
sets number of enabled sensors with newNumEnabledOfSer	
sets maximum number of sensors with newMaxNumOfSensc	

1.2.5 CameraManagement



1. Description

This class is for camera management. All functions related to camera management are accomplished in this class. User accessing internet can turn on or off certain cameras, and add or delete cameras. And also user can modify maximum number of cameras.

2. Attribute

1. **maxNumOfCameras**: maximum number of cameras that can exist in the house.
2. **numOfEnabledCameras**: current number of enabled cameras.
3. **cameraList**: all camera list in the house.
4. **EnabledCameraList**: enabled camera list.
5. **malfunctioningCameraList**: list of malfunctioning cameras.
6. **ignoredCameraList**: list of ignored cameras.
7. **videoList** : list of videos.
8. **maxNumOfVideo:** maximum number of videos.

3. Responsibilities

1. **bool initialize()**
   - Precondition: CoreContol's initialize() function is called.
   - Postcondition: Cameras are initialized.
   - Trigger: When user turns on the power of safeHome system.
     When panel is "away" activated.
   - This function is used when safeHome power turns on. And whenever the system is "away" or "stay" activated, CameraManagement's initialize() function must be called. If this function is properly done, and cameras are properly initialized, true is returned.

2. **bool enableAll()**
   - Precondition: SafeHome is "away" or "stay" activated.
   - Postcondition: All cameras are enabled.
   - Trigger: When SafeHome is "away" or "stay" activated.
   - This function is used when safeHome is "away" or "stay" activated. And whenever the system is "away" or "stay" activated, CameraManagement's enables all cameras in the house. So this function calls all cameras' enable() function. If this function is properly done, true is returned.

3. **bool disableAll()**
   - Precondition: SafeHome is deactivated.
   - Postcondition: All cameras are disabled.
   - Trigger: When SafeHome is deactivated.
   - This function is used when safeHome is deactivated. And whenever the system is deactivated, CameraManagement's disables all cameras in the house. So this function calls all cameras' disable() function. If this function is properly done, true is returned.

4. **bool enable(cameraID)**
   - Precondition: User accessing via web interface enables camera whose id is cameraID.
   - Postcondition: camera which has cameraID is enabled.
   - Trigger: When user accessing via web interface enables camera whose id is cameraID.
   - This function is used when safeHome is "away" or "stay" activated and user accessing via web interface enables certain camera whose id is cameraID. So this function calls the selected camera's enable() function. If this function is properly done, true is returned.

5. **bool disable(cameraID)**
   - Precondition: User accessing via web interface disables sensor whose id is cameraID.
   - Postcondition: camera which has cameraID is disabled.
   - Trigger: When user accessing via web interface disables camera whose id is cameraID.
   - This function is used when safeHome is "away" or "stay" activated and user accessing via web interface disables certain camera whose id is cameraID. So this function calls the selected camera's disable() function. If this function is properly done, true is returned.

6. **Camera searchCamera(String cameraID)**
   - Precondition: None
   - Postcondition: Camera which has cameraID is returned.
   - Trigger: When functions which needs to search certain camera with cameraID calls this function.
   - This function is used when functions which need to search certain camera with cameraID are called. If this function is properly done, sensor whose id is cameraID is returned.

7. **bool addCamera(String cameraID, String type)**
   - Precondition: User accessing vie web interface requests to add new camera which has cameraID and type to cameraList.
   - Postcondition: A new camera is added to cameraList.
   - Trigger: When user accessing vie web interface requests to add new camera which has cameraID and type.
   - This function used when user accessing via web interface requests to add new camera. If this function is properly done, camera whose id is cameraID and type is type is added to cameraList. A newly added camera is initially diabled. And if this function is properly done, camera whose id is cameraID and type is type is added.

8. **Camera deleteCamera(String cameraID)**
   - Precondition: User accessing vie web interface requests to delete camera which has cameraID from cameraList.
   - Postcondition: The selected camera is deleted form cameraList.
   - Trigger: When user accessing vie web interface requests to delete camera which has cameraID from cameraList.
   - This function used when user accessing via web interface requests to delete camera whose id is cameraID. If this function is properly done, camera is deleted from cameraList. And if this function is properly done, camera whose id is cameraID returned.

9. **bool addMalfunctioningCamera(Camera c)**
   - Precondition: None.
   - Postcondition: A malfunctioning camera is added to malfunctioningCameraList.
   - Trigger: While accomplishing any functions related to cameras, if one camera is malfunctioning, this function is called.
   - If malfunctioning camera is found, the camera is added to malfunctioningCameraList. And if this function is properly done, true is returned.

10. **Camera deleteMalfunctioningCamera(Camera c)**
    - Precondition: None.
    - Postcondition: Sensor s is deleted from malfunctioningCameraList.
    - Trigger: When malfunctioning camera is fixed or trashed.
    - This function deletes Camera c from the malfunctioningCameraList. And if this function is properly done, the deleted camera is returned.

11. **bool addIgnoredCamera(Camera c)**
    - Precondition: Panel is deactivated and Initialize() function is conducted.
    - Postcondition: Camera c which should be ignored is added to ignoredCameraList.

- Trigger: When panel is deactivated, if there is sensor which is not disabled by any reasons.
- When panel is deactivated, if there is camera which is not disabled by any reasons. But camera must be forcibly ignored. So the camera must be added to ignoredCameraList and safeHome system ignores all signals from cameras in ignoredCameraList. And if this function is properly done, true is returned.

12. **Camera deleteIgnoredCamera(Camera c)**
    - Precondition: None
    - Postcondition: Camera c is deleted from ignoredCameraList.
    - Trigger: indecisive
    - This function deletes Camera c from the ignoredCameraList. And if this function is properly done, the deleted Camera is returned.

13. **Image requestImage(cameraID)**
    - Precondition: User accessing via web interface requests image of camera whose id is cameraID.
    - Postcondition: image of camera is returned.
    - Trigger: When user accessing via web interface requests image of camera whose id is cameraID.
    - This function returns image of camera whose id is cameraID. And if this function is properly done, the Image is returned.

14. **bool zoom_in(cameraID)**
    - Precondition: User accessing via web interface requests to zoom_in function of camera whose id is cameraID.
    - Postcondition: image is zoomed in.
    - Trigger: When user accessing via web interface requests to zoom_in function of camera whose id is cameraID.
    - This function returns true if the zoom_in function of the camera is conducted.

15. **bool zoom_out(cameraID)**
    - Precondition: User accessing via web interface requests to zoom_out function of camera whose id is cameraID.
    - Postcondition: image is zoomed out.
    - Trigger: When user accessing via web interface requests to zoom_out function of camera whose id is cameraID.
    - This function returns true if the zoom_out function of the camera is conducted.

16. **bool pan(cameraID)**
    - Precondition: User accessing via web interface requests to pan function of camera whose id is cameraID.
    - Postcondition: image pans.
    - Trigger: When user accessing via web interface requests to pan function of camera whose id is cameraID.
    - This function returns true if the pan function of the camera is conducted.

17. **bool tilt(cameraID)**
    - Precondition: User accessing via web interface requests to tilt function of camera

whose id is cameraID.
- Postcondition: image tilts.
- Trigger: When user accessing via web interface requests to tilt function of camera whose id is cameraID.
- This function returns true if the tilt function of the camera is conducted.

18. **Image makeThumbnailImage()**
    - Precondition: User accessing via web interface requests thumbnail image.
    - Postcondition: ThmbnailImage is returned.
    - Trigger: When user accessing via web interface requests thumbnail image.
    - This function makes thumbnail image of all cameras by using each camera's scanImage() function. And this function returns thumbnail image.

19. **String cameraListInfo()**
    - Precondition: When user accessing via web interface requests cameraList information.
    - Postcondition: cameraList information is made and returned.
    - Trigger: When user accessing via web interface requests cameraList information.
    - This function makes String which contains information about cameras which is in cameraList. And if this function is properly done, the String which made previously is returned.

20. **String malfunctioningCameraListInfo()**
    - Precondition: None
    - Postcondition: malfunctioningCameraList information is made and returned.
    - Trigger: indecisive
    - This function makes String which contains information about cameras which is in malfunctioningCameraList. And if this function is properly done, the String which made previously is returned.

21. **String ignoredCameraListInfo()**
    - Precondition: None
    - Postcondition: ignoredCameraList information is made and returned.
    - Trigger: indecisive
    - This function makes String which contains information about cameras which are in ignoredCameraList. And if this function is properly done, the String which made previously is returned.

22. **bool record(String cameraID)**
    - Precondition: User requests to start recording a view of the selected camera.
    - Postcondition: The camera recording is started.
    - Trigger: When User requests to start recording a view of the selected camera.
    - This function records view of the selected camera. If the starting to record is properly done, this function returns true.

23. **videoFile stopRecord(String cameraID)**
    - Precondition: User requests to end recording a view of the selected camera.
    - Postcondition: The camera recording is ended.
    - Trigger: When User requests to end recording a view of the selected camera.

- This function records view of the selected camera. If the stopping to record is properly done, this function returns the video which is recorded by previous procedure.

24. **bool setMaxNumOfCameras(int newMaxNumOfCameras)**
    -Precondition: User requests to modify the maximum number of cameras in the house.
    - Postcondition: The maximum number of cameras is modified.
    - Trigger: When User requests to modify the maximum number of cameras.
    -This function modifies maximum number of cameras. This function returns true if maximum number of cameras is properly changed.
25. **void setMaxNumOfVideo(int newMaxNumOfVideos)**
    - Precondition: User requests to modify the maximum number of videos.
    - Postcondition: The maximum number of videos is modified.
    - Trigger: When User requests to modify the maximum number of videos.
    -This function modifies maximum number of videos.

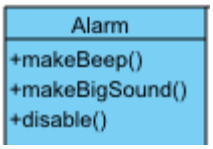
4. Traceability

Responsibilities	**Use Case Description**
+bool initialize()	3.1 Initialization
+bool enableAll()	3.2 Activation, 3.6 Access Cameras
+bool disableAll()	3.3 Deactivation, 3.6 Access Cameras
+bool enable(cameraID)	3.6 Access Cameras
+bool disable(cameraID)	3.6 Access Cameras
+bool addMalfunctioningCamera(Camera c)	3.1 Initialization, 3.2 Activation
+bool addIgnoredCamera(Camera c)	3.2 Activation
+Image requestImage(cameraID)	3.6 Access Cameras
+bool zoom_in(cameraID)	3.6 Access Cameras
+bool zoom_out(cameraID)	3.6 Access Cameras
+bool pan(cameraID)	3.6 Access Cameras
+bool tilt(cameraID)	3.6 Access Cameras
+Image makeThumbnailImage()	3.6 Access Cameras
+bool record(String cameraID)	Newly added
+ImgFile stopRecord(String cameraID)	Newly added

5. CRC Card

CameraManagement	
Super Classes:	
Sub Classes:	
Description: CameraManagementModule deals with thouse command as well as manages all cameras in the house.	
Attributes:	
Name	Description
numOfEnabledCameras	number of enabled cameras
maxNumOfCameras	maximum number of cameras
cameraList	list of cameras which is currently enabled
enabledCameraList	list of enabled camera
malfunctioningCameraList	list of cameras which is malfunctioning
ignoredCameraList	list of cameras which is ignored
videoList	list of stored video
maxNumOfVideo	maximum number of videos which is sotred
Responsibilities:	
Name	Collaborator
initializes all cameras	Camera
enables all cameras at once	Camera
disables all cameras at once	Camera
enables camera which has the cameraID	Camera
disable camera which has the cameraID	Camera
searches the camera with the cameraID	
adds camera to cameraList with the cameraID and type	
deletes camera from cameraLlst with the cameraID	
adds malfunctioning camera to malfuctioningCameraList with t	
deletes malfunctioning camera from malfunctioningCameraList	
adds ignored camera to ignoredCameraList with the camera ob	
deletes ignored camera from ignoredCameraList with the came	
requests images from the camera which has cameraID	
zooms in the camera which has cameraID	Camera
zooms out the camera which has cameraID	Camera
pans the camera which has cameraID	Camera
tilts the camera which has cameraID	Camera
makes thumbnail camera image	
makes string about cameraList	Camera
makes string about malfunctioningCameraList	Camera
makes string about ignoredCameraList	Camera
records the view of the camera which has cameraID	Camera
stops the recording of the camera which has cameraID	Camera
sets max number of cameras with newMaxNumOfCameras	
sets max number of videos with newMaxNumOfVideos	
adds enabled camera to enabledCameraList	
deletes deletes enabled from enabledCameraList	

1.2.6 Alarm



1. Description

This is Alarm class. This class is mainly responsible for make a big alarming sound in emergency situation, but also it is concerned about a small beep sound when needed.

2. Attribute

None

3. Responsibilities

1. **void makeBeep()**
   - Precondition: Something not expected has been happened.
   - Postcondition: A short beep sound is played.
   - Trigger: When something not expected has been happened.
   - When a small beep sound is needed, this function makes a small beep sound to notify the user that something is wrong.
2. **void makeBigSound()**
   - Precondition: System concluded that it is an emergency situation.
   - Postcondition: A big alarm sound is played.
   - Trigger: When system concluded that it is an emergency situation.
   - When system detected motion while activated, or an intruder tries to enter, or user pressed panic button, or in any other emergency situations, system should alarm. This function takes responsibility for making Big alarm sound in such emergency situations.
3. **void disable()**
   - Precondition: User wants to turn off the alarm.
   - Postcondition: Alarm is disabled.
   - Trigger: When User wants to turn off the alarm.
   - Sometimes system might make incorrect decision and user wants to turn off the alarm. Or, after user solved the problem which is a reason for alarming user may want to turn off the alarm. In such situations, this function disables alarm.

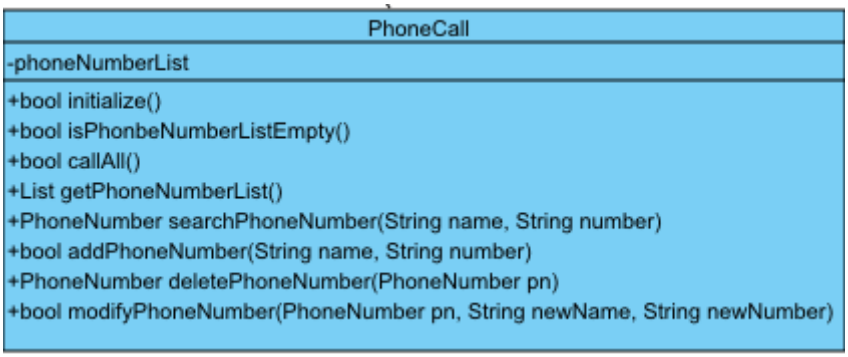
4. Traceability

Responsibilities	**Use Case Description**
+void makeBeep()	Newly added
+void detect()	3.7 Alert
+void disable()	3.3 Deactivation

5. CRC Card

Alarm	
Super Classes:	
Sub Classes:	
Description: When intruder detected in the house or tried unallowed access, the safeHome system alert with alaram sound.	
Attributes:	
Name	Description
Responsibilities:	
Name	Collaborator
makes beep sound(small notice sound)	
makes big sound(bic alarm sound)	
disables alram	

1.2.7 Phonecall



1. Description

This class is for phone call. All functions related to phone call are accomplished in this class.

2. Attribute

1. **phoneNumberList**: list of phone numbers which is called in emergency situation.

3. Responsibilities

1. **bool initialize()**
   - Precondition: Phonecall class is needed to be initialized.
   - Postcondition: Phonecall class is initialized.
   - Trigger: When Phonecall class is needed to be initialized.
   - When Phonecall class is needed to be initialized, (i.e. Activation or Initialization) this function initializes the Phonecall class and returns the result. If initialization was successful, this function returns true. Otherwise, it returns false.
2. **bool isPhoneNumberListEmtpy()**
   - Precondition: None.
   - Postcondition: Returns true if phoneNumberList is empty. Otherwise returns false.
   - Trigger: indecisive.
   - This function returns true if phoneNumberList is empty. If phoneNumberList is not empty, it returns false.
3. **bool callAll()**
   - Precondition: In emergency situations.
   - Postcondition: Call to the all phone numbers in phoneNumberList.
   - Trigger: In emergency situations.
   - In emergency situations, the call to all the phone numbers in phoneNumberList may be needed. This function tries to call to all the phone numbers in phoneNumberList, and returns true if all the calls are successful. If any call has been failed, this returns false.
4. **List getPhoneNumberList()**
   - Precondition: Other classes need information about phone numbers.
   - Postcondition: Returns List which is identical to phoneNumberList.
   - Trigger: When other classes need information about phone numbers.
   - When other class need phoneNumberList, this function confirms the identical List to the phoneNumberList and returns it.

5. **PhoneNumber searchPhoneNumber(String name, String number)**
   - Precondition: None.
   - Postcondition: Returns PhoneNumber which matches name and number.
   - Trigger: indecisive.
   - This function finds specific PhoneNumber instance in phoneNumberList which matches name and number passed by arguments and returns it.
6. **PhoneNumber searchPhoneNumber(String name, String number)**
   - Precondition: None.
   - Postcondition: Returns PhoneNumber which matches name and number.
   - Trigger: indecisive.
   - This function finds specific PhoneNumber instance in phoneNumberList which matches name and number passed by arguments and returns it.

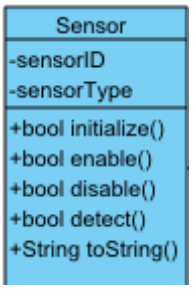
4. Traceability

Responsibilities	**Use Case Description**
+bool initialize()	3.1 Initialization, 3.2 Activation
+bool phoneNumberListIsEmpty()	3.8 Configuration
+bool callAll()	3.7 Alert
+List getPhoneNumberList()	3.8 Configuration
+PhoneNumber searchPhoneNumber(String name, String number)	3.8 Configuration
+bool addPhoneNumber(String name, String number)	3.8 Configuration
+PhoneNumber deletePhoneNumber(PhoneNumber pn)	3.8 Configuration
+bool modifyPhoneNumber(PhoneNumber pn, String newName, String newNumber)	3.8 Configuration

5. CRC Card

PhoneCall	
Super Classes:	
Sub Classes:	
Description: PhoneCall class accomplishes functions which related to phonecall module. When emerg	
Attributes:	
Name	Description
phoneNumberList	list of phone number
Responsibilities:	
Name	Collaborator
initializes phoneCall module	
checks if PhoneNumberList is empty or not	
makes calls to all numbers in the phoneNumberList	
searches PhoneNumber with name and number	
adds PhoneNumber with name and number	
deletes phoneNumber with PhoneNumber object	

1.2.8 Sensor



Figure

1. Description

This class is Sensor class. This class is designed for control sensors one by one. Each class is responsible for exact one sensor. It has all functions about sensors; initialization, enable/disable, and detection.

2. Attribute

1. **sensorID**: ID for the sensor.
2. **sensorType**: Specify type of the sensor.

3. Responsibilities

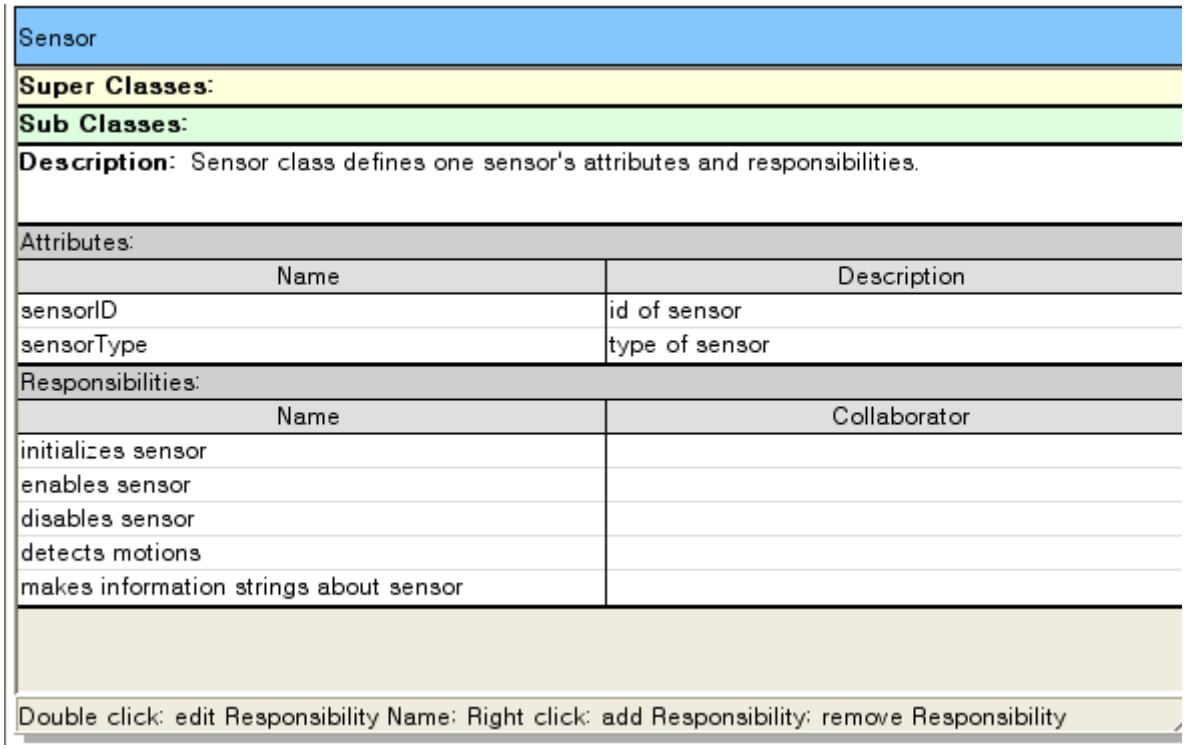
1. **bool initialize()**
   - Precondition: Sensor is needed to be initialized.
   - Postcondition: Sensor is initialized.
   - Trigger: When Sensor is needed to be initialized.
   - When Sensor is needed to be initialized, (i.e. Activation or Initialization) this function initializes the Sensor and returns the result. If initialization was successful, this function returns true. Otherwise, it returns false.
2. **bool enable()**
   - Precondition: Sensor is needed to be enabled.
   - Postcondition: Sensor is enabled.
   - Trigger: When Sensor is needed to be enabled.
   - When Sensor is needed to be enabled, (i.e. Activation or Configuration) this function enables the Sensor and returns the result. If the sensor is successfully enabled, this function returns true. Otherwise, it returns false.
3. **bool disable()**
   - Precondition: Sensor is needed to be disabled.
   - Postcondition: Sensor is disabled.
   - Trigger: When Sensor is needed to be disabled.
   - When Sensor is needed to be disabled, (i.e. Deactivation or Configuration) this function disables the Sensor and returns the result. If the sensor is successfully disabled, this function returns true. Otherwise, it returns false.
4. **bool detect()**
   - Precondition: None
   - Postcondition: Returns true if some motions has been detected.
   - Trigger: indecisive.
   - If some motion has been detected, this function returns true. If no motion has been detected, this function returns false.

5. **String toString()**
   - Precondition: None
   - Postcondition: Returns string that contains information about the sensor.
   - Trigger: indecisive.
   - This function returns string that contains information about the sensor.

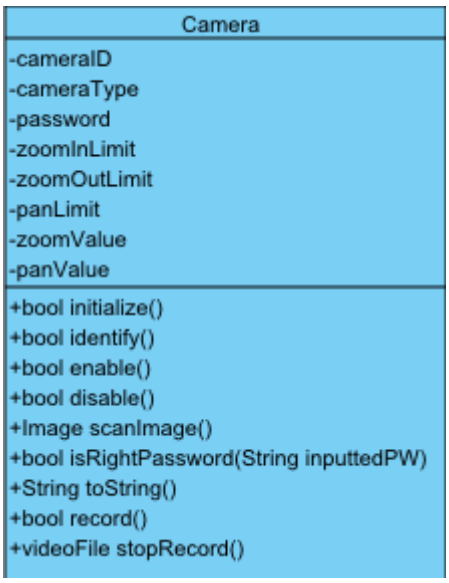
4. Traceability

Responsibilities	**Use Case Description**
+bool initialize()	3.1 Initialization
+bool enable()	3.2 Activation
+bool disable()	3.3 Deactivation
+bool detect()	3.7 Alert
+String toString()	Newly added

5. CRC Card



1.1.9 Camera



Figure

1. Description

This is Camera class. This class is designed for control cameras one by one. Each class is responsible for exact one camera. It has all functions about cameras; initialization, enable/disable, identification, scanning image, password verification, and recording.

2. Attribute

1. **cameraID**: ID for the camera.
2. **cameraType**: Type of the camera.
3. **password**: password which is needed when requesting view.
4. **zoomInLimit**: Limit that camera can zoom in.
5. **zoomOutLimit**: Limit that camera can zoom out.
6. **panLimit**: Limit that camera can pan.
7. **zoomValue**: Current Value that represents how much the camera has zoomed in.
8. **panValue**: Current value that represents location of the camera.

3. Responsibilities

1. **bool initialize()**
   - Precondition: Camera is needed to be initialized.
   - Postcondition: Camera is initialized.
   - Trigger: When Camera is needed to be initialized.
   - When Camera is needed to be initialized, (i.e. Activation or Initialization) this function initializes the Camera and returns the result. If initialization was successful, this function returns true. Otherwise, it returns false.

2. **bool enable()**
   - Precondition: Camera is needed to be enabled.
   - Postcondition: Camera is enabled.
   - Trigger: When Camera is needed to be enabled.
   - When Camera is needed to be enabled, (i.e. Activation or Configuration) this

function enables the Camera and returns the result. If the Camera is successfully enabled, this function returns true. Otherwise, it returns false.

3. **bool disable()**
   - Precondition: Camera is needed to be disabled.
   - Postcondition: Camera is disabled.
   - Trigger: When Camera is needed to be disabled.
   - When Camera is needed to be disabled, (i.e. Deactivation or Configuration) this function disables the Camera and returns the result. If the Camera is successfully disabled, this function returns true. Otherwise, it returns false.

4. **Image scanImage()**
   - Precondition: User accessing via web interface requests image of camera
   - Postcondition: image of camera is returned.
   - Trigger: When user accessing via web interface requests image of camera.
   - This function scans and returns image of camera. And if this function is properly done, the Image is returned.

5. **bool isRightPassword(String inputtedPW)**
   - Precondition: User inputted password for accessing camera view.
   - Postcondition: Return true if password is correct.
   - Trigger: When user inputted password for accessing camera view.
   - If user wants to access camera view, user must enter the correct password which is pre-configured for each camera. This function compares password that user has inputted with password that is stored in the class. And if password is correct, the function returns true. If password is incorrect, the function returns false.

6. **String toString()**
   - Precondition: None
   - Postcondition: Returns string that contains information about the camera.
   - Trigger: indecisive.
   - This function returns string that contains information about the camera.

7. **bool record()**
   - Precondition: User requests to start recording a view of the camera.
   - Postcondition: Recording is started.
   - Trigger: When User requests to start recording a view of the selected camera.
   - This function records view of the camera. If the starting to record is properly done, this function returns true.

.

8. **videoFile stopRecord()**
   - Precondition: User requests to end recording a view camera.
   - Postcondition: Recording is ended.
   - Trigger: When User requests to end recording a view of the camera.
   - This function records view of the camera. If the stopping to record is properly done, this function returns the video which is recorded by previous procedure.

4. Traceability

Responsibilities	Use Case Description
+bool initialize()	3.1 Initialization
+bool identify()	3.1 Initialization, Access Camera
+bool enable()	3.2 Activation
+bool disable()	3.3 Deactivation
+Image scanImage()	3.6 Access Cameras
+bool IsRightPassword(String inputtedPW)	3.6 Access Cameras
+String toString()	Newly added
+bool record()	3.6 Access Cameras
+imgFile stopRecord()	3.6 Access Cameras

5. CRC Card

Camera

Super Classes:

Sub Classes:

Description: Camera class defines one camera's attributes and responsibilities.

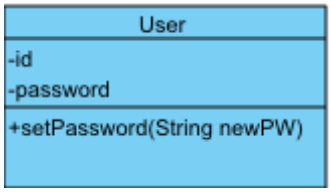
Attributes:

Name	Description
cameraID	id of camera
cameraType	type of camera
password	password of camera
zoomInLimit	zoom in limit of camera
zoomOutLimit	zoom out limit of camera
panLimit	pan limit of camera
zoomValue	value of zoom rate
panValue	value of pan

Responsibilities:

Name	Collaborator
initializes camera	
identifies camera	
enables camera	
disables camera	
scans camera current image	
checks that pasword which is inputted by user is	
makes information strings about camera	
starts to record	
stops recording	

1.2.10 User



1. Description

This class is about user who accesses via internet. User has id and password.

2. Attribute

1. **id**: User's id on the web.
2. **password**: User's password on the web.

3. Responsibilities

1. setPassword(String new PW)

- Precondition: User must logs in.
- Postcondition: User's password is changed.
- Trigger: User requests to change password.
- This function changes user's password with new password.

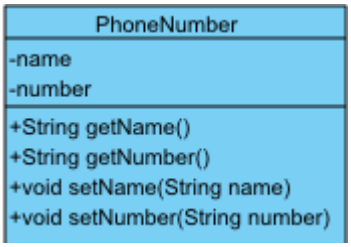
4. Traceability

Responsibilities	**Use Case description**
setPassword(String newPW)	3.8 Configuration

5. CRC Card

User	
Super Classes:	
Sub Classes:	
Description: User class defines users who access via internet with id and password	
Attributes:	
Name	Description
id	id of user who is access via internet
password	password of user
Responsibilities:	
Name	Collaborator
sets password with new password	

1.2.11 PhoneNumber



Figure

1. Description

This class about PhoneNumber. Phone number has name and number in it. PhoneNumber is needed to encapsulate phone number object.

2. Attribute

1. **name**: Owner of the number
2. **number**: phone number

3. Responsibilities

1. String getName()

- Precondition: None
- Postcondition: name is returned.
- Trigger: indecisive
- This function returns the name.

2. String getNumber()

- Precondition: None
- Postcondition: number is returned.
- Trigger: indecisive
- This function returns the number.

3. void setName(String name)

- Precondition: None
- Postcondition: name is modified.
- Trigger: indecisive
- This function modifies name.

4. void setNumber(String number)

- Precondition: None
- Postcondition: number is modified.
- Trigger: indecisive
- This function modifies number.

4. Traceability

- In Use case description there is no concrete description about PhoneNumber

5 CRC Card

PhoneNumber	
Super Classes:	
Sub Classes:	
Description: PhoneNumber class defines phone number	
Attributes:	
Name	Description
name	name who receives emergency call
number	phone number
Responsibilities:	
Name	Collaborator
gets name	
gets number	
sets name with a new name	
sets number with a new number	

1.2.12 FloorPlan

Figure

1. **Description**

This class is for floor-plan. The superior class of floor-plan. This class have the list of floor. When other classes request the floor plans or set up floor plans, response properly.

2. **Attribute**
   1. numberOfFloor : The number of floor that this plan has.
   2. floorList : the list of floor.

3. **Responsibilities**
   1. getFloorPlan()

- Precondition: Stored floor-plans exist.
- Postcondition: Floor plan will be returned.
- Trigger: Some functions wants to get the floor-plan.
- This function is for get the entire data.

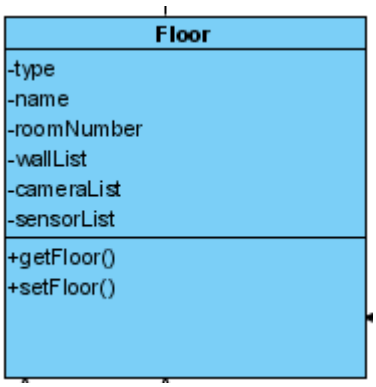
   2. setFloorPlan()

- Precondition: The storing space is available.
- Postcondition: New of modified floor-plan will be stored.
- Trigger: Some functions wants to set the floor-plan.(Maybe, configuration)
- This function is for set or modifying the entire data.

4. Traceability

- In Use case description there is no concrete description about Floor Plan class

1.2.13 Floor



1. Description

This is Floor class. One floor-plan can store several floors. This class represents one floor. This floor has cameras, sensors and walls.

2. Attribute

1. type : The type of floor.
2. name : The name of floor.
3. roomNumber : How many rooms the floor have.
4. wallList: The list of walls that this floor have
5. cameraList: The list of cameras that this floor have
6. sensorList: The list of sensors that this floor have

3. Responsibilities

1. getFloor()

- Precondition: Stored floor exist.
- Postcondition: Floor information will be returned.
- Trigger: Some functions wants to get the floor-plan. And the floor-plan class wants to get the floor.
- This function is for get the entire data of one floor.

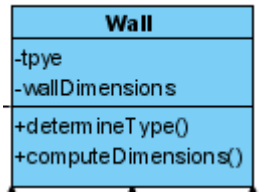
2. setFloor()

- Precondition: The storing space is available.
- Postcondition: New of modified floor will be stored.
- Trigger: Some functions wants to set the floor-plan.(Maybe, configuration) And the floor-plan class wants to set the floor.
- This function is for set or modifying the entire data of one floor.

4. Traceability

- In Use case description there is no concrete description about Floor class

1.2.14 Wall



1. Description

This is CoreControl class. This class represents the core of system. This class is the superior class than Safehome's functionality classes. Other functionality classes, sensor management, camera management and so on, connected from this class. And this CoreControl class orders the other class.

2. Attribute

1. type : the type of wall

3. Responsibilities

1. getWall()

- Precondition: Stored walls exist.
- Postcondition: Wall information will be returned.
- Trigger: Some functions wants to get the floor-plan. And the floor-plan class wants to get the floor. And next the floor is needed. Finally, these walls need to return.
- This function is for get the entire data of one wall.

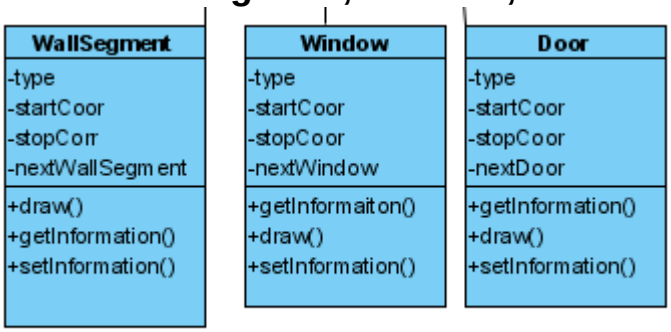
2. setWall()

- Precondition: The storing space is available. And the floor that this wall is belonged has no error.
- Postcondition: New of modified wall will be stored.
- Trigger: Some functions wants to set the floor-plan.(Maybe, configuration). Setting floor is the set of setting walls.

4. Traceability

- In Use case description there is no concrete description about Wall class

1.2.15 Wall segment, Window, Door



1. **Description**

   Walls have wall segments, windows and doors. Each component is object. So all the components are from classes. And these classes are sub-classes from the super class, Wall class. The three classes in abstract level are not quite different and have same attributes and responsibility.

2. **Attribute**
   1. type
   2. startCoor : the coordinate of starting point.
   3. stopCoor : the coordinate of ending point.
   4. next… : If there is connected Component, next… is indicated the next componets.

3. **Responsibilities**

   1. getInformation()

   - Precondition: Stored walls exist.
   - Postcondition: Wall information will be returned.
   - Trigger: Some functions wants to get the floor-plan. And the floor-plan class wants to get the floor. And next the floor is needed. Finally, these wall information need to return.
   - This function is for get the entire data of one wall.

   2. setInformation()

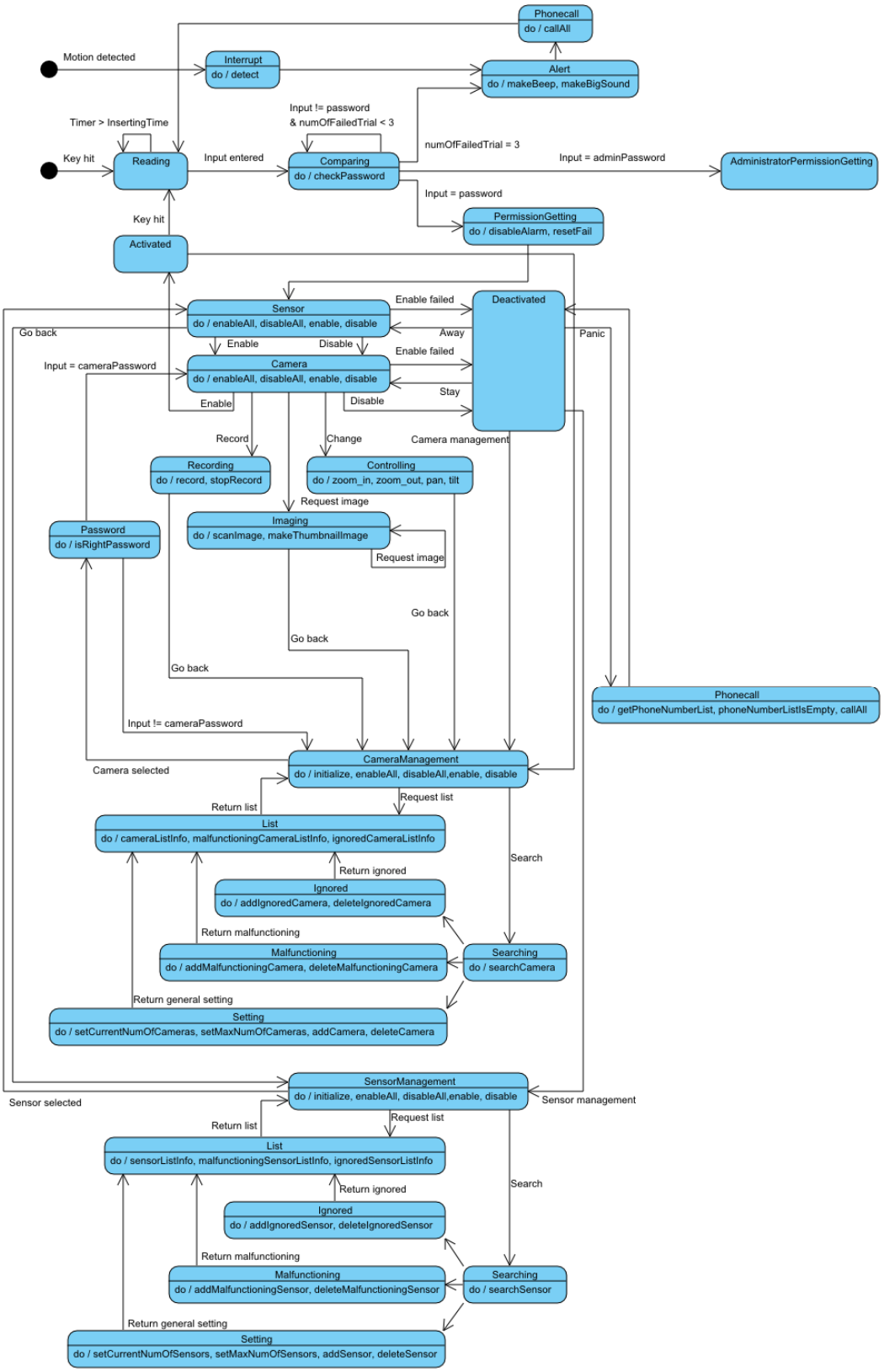
   - Precondition: The storing space is available. And the wall that these wall components are belonged hae no error.
   - Postcondition: New of modified wall components will be stored.
   - Trigger: Some functions wants to set the floor-plan.(Maybe, configuration). Setting floor is the set of setting walls.

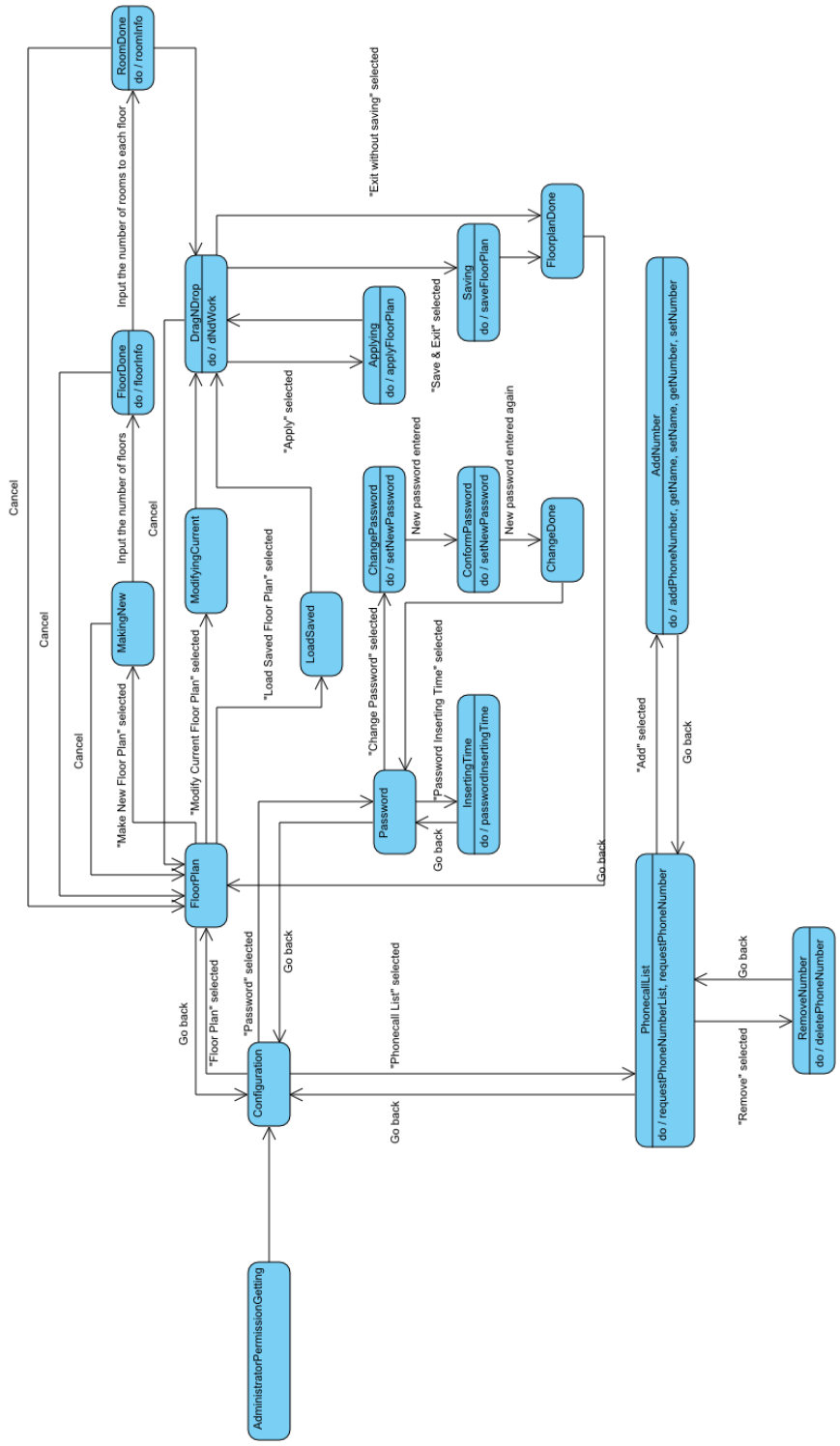
4. Traceability

- In Use case description there is no concrete description

2. State diagram

2.0 The entire view





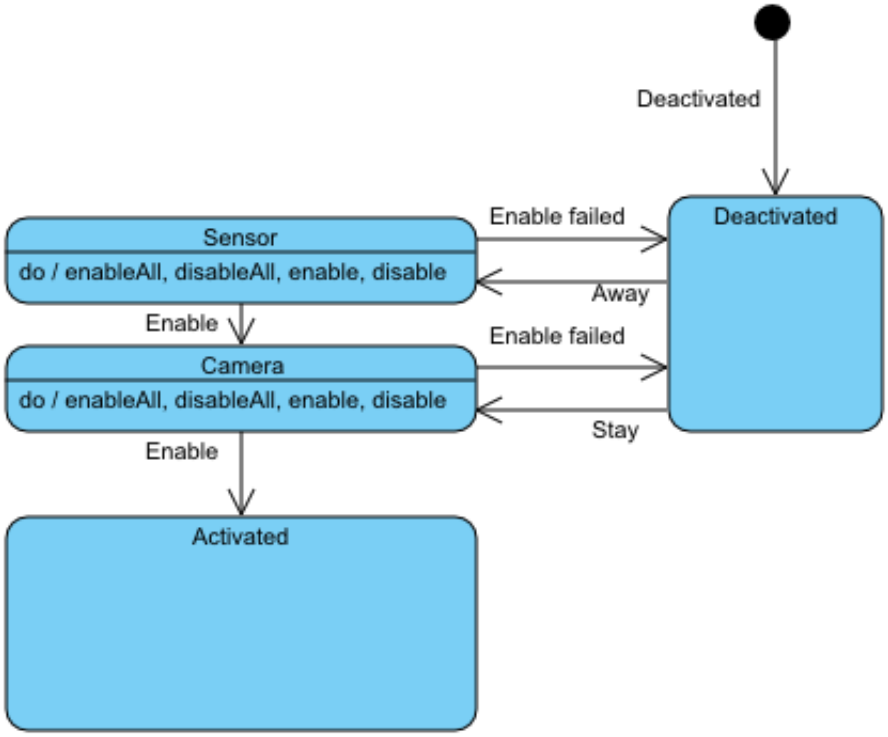
Descprion

Starting with key hit, states flow. Every function needs qualification to access the system. So, starting state goes to permission process and then user deactivates the system or administrator changes configuration.

Most states have “Go back” transition to go to previous state, and some states have “Cancel” transition to cancel current work.

Not only key hit but also detected motion by sensors can be the stating state

2.1 Activate



Description

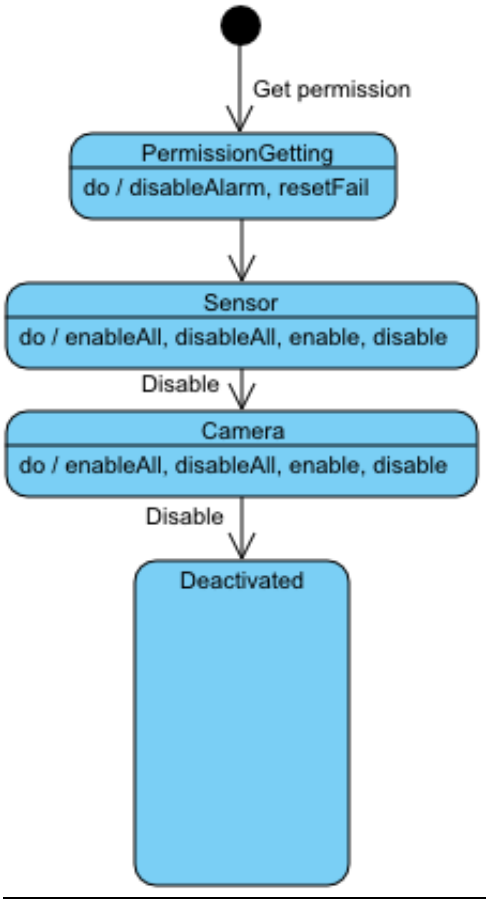
The main purpose of Safehome system is security. The main function of security is this “Activation.” User turns on the sensors and cameras of system. This diagram is for only “Activation” use-case. The use-case is described in the use-case statement.

This diagram described how the system state is changed when the system is activated. There is two activation states, stay and away.

State Flow

This diagram starts at the deactivated state. When the user want to activate “stay” or “away” mode, the state is changed to activation state. On the way that sensor and camera is turned on, if one of them is failed (can not be enabled), the state is returned to deactivate state.

2.2 Deactivate



Description

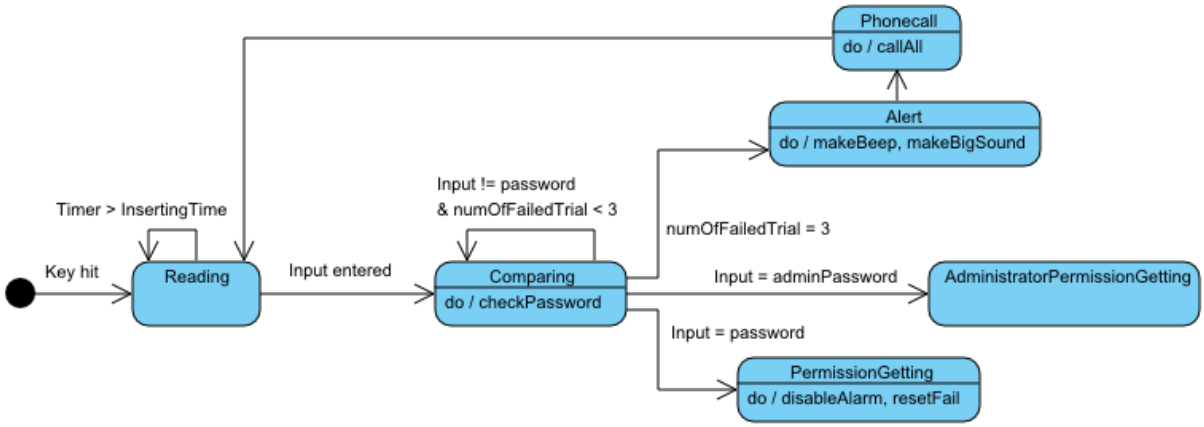
When the system is activated, all the people in the house is detected by the sensors. But when he or she input the password shortly, the system is deactivated and the Safehome system doesn't regard a user as an intruder. This diagram is for only "Deactivation" use-case. The use-case is described in the use-case statement.

This diagram described how the system state is changed when the system is deactivated.

State Flow

At first, user must get permission. If user gets the permission, user must be not an intruder. So the activated state will be changed. The all the security components will be stopped. If the entire component is deactivated and stopped, the system's status is "deactivated"

2.3
Permission



Description

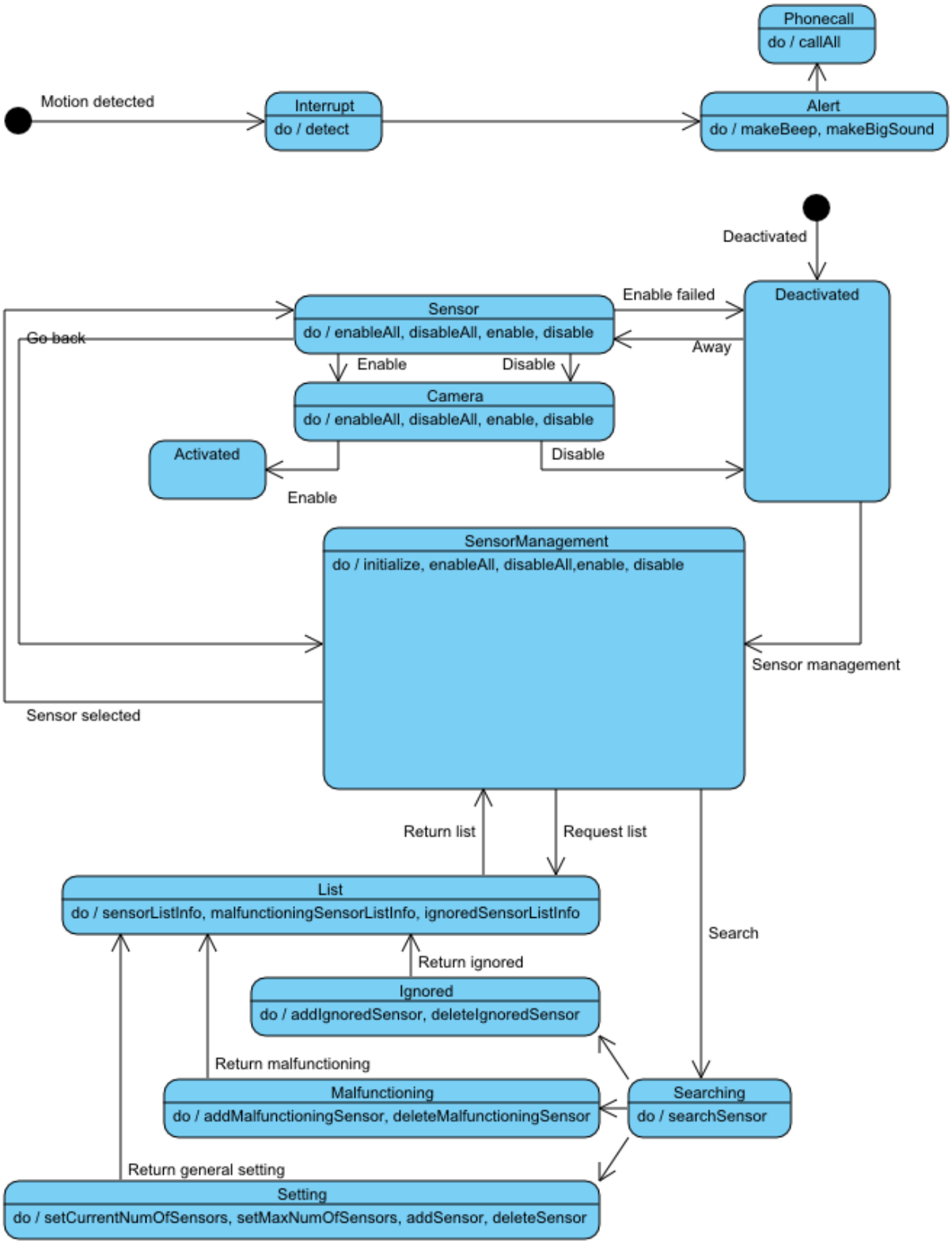
All users must to have the qualification to access the system. This qualification is for deactivating the system in our SafeHome. User can get permission by inputting 4-digit password, and user has 3 chances for correct password. And there is password for administrator. Only administrator can change configuration and get administrator permission only by administrator password.

State Flow

Permission starts from any key hit. Basic state of Permission is Reading. In this state, every input is considered as password entered. If user inputs correct password, then user gets permission directly. Also, there is inserting time between each character. If the time between characters goes over inserting time, inputs which have been entered will be gone and next input will be considered as the first input.

Whenever user entered incorrect password, numOfFailedTrial increases by 1. If it reaches 3, Alert is performed. If user enters correct password in any step after Alert is performed, that makes Alert disable and numOfFailedTrial is reset to 0. Also, numOfFailedTrial is reset to 0 after user gets permission.

2.4 Access Sensors



Description

User can modify the state of sensors via internet. In Activation or Deactivation module controls all the sensors at a time. But user can turn on or off each sensor.

State Flow

If unexpected motion is detected by sensors, current state is changed to interrupt state and Alert is performed.

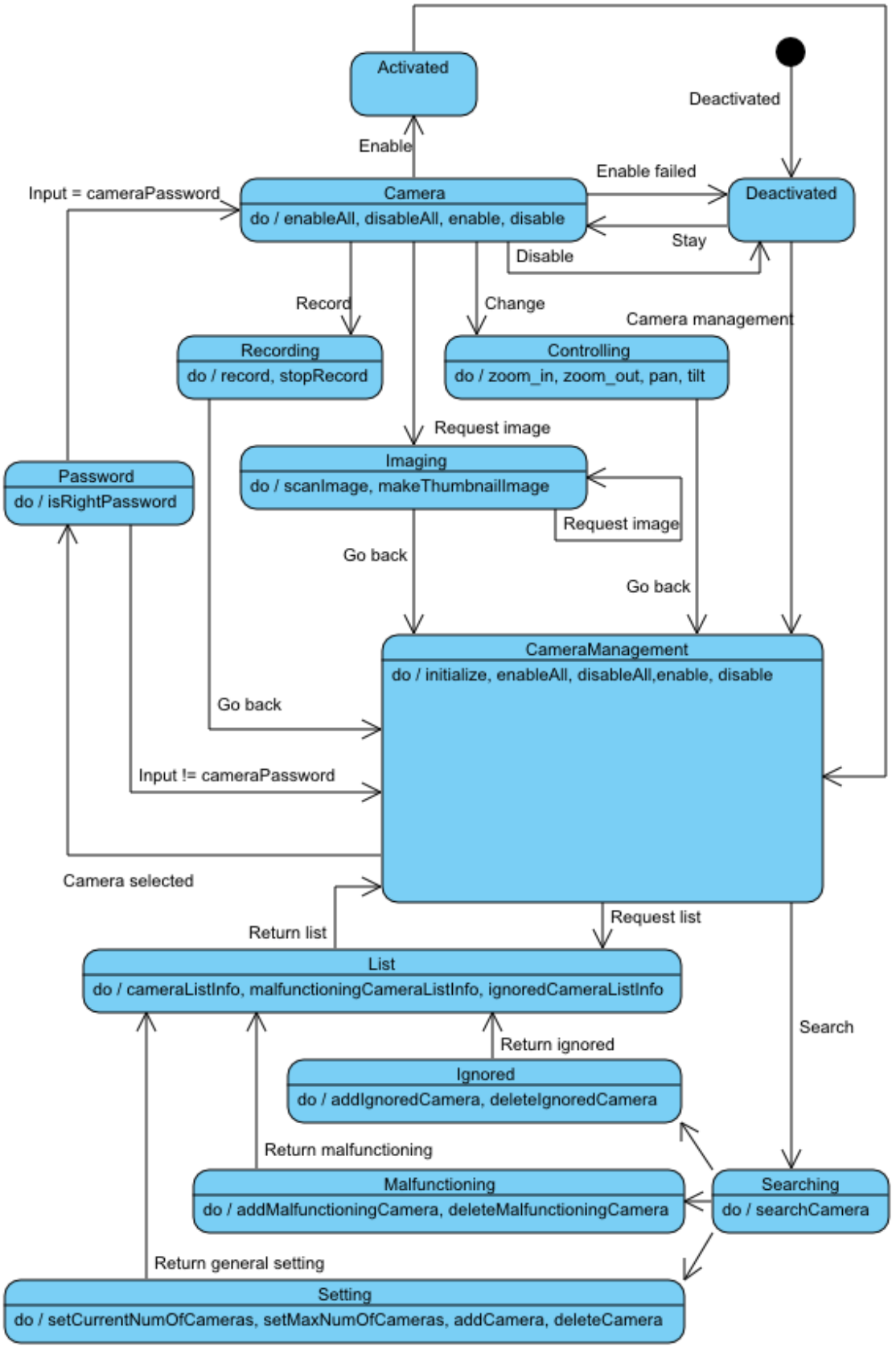
To turn off each sensor, user selects a sensor and makes it disable. User, also, can turn off all sensors by using disableAll function. To turn on sensors, user uses enable function and enableAll function in same way.

With search function, user can search all sensors and find sensors which are malfunctioning. After searching is done, the list of the malfunctioning sensors is returned to the system, and user can see the list.

Also, user can ignore malfunctioning sensors, and put them on the list of the ignored sensors. User can add or remove sensors and make the list of them with this management, too.

If user requests the list of any type of sensors directly, the information of sensors sensorListInfo, malfunctioningSensorListInfo, ignoreSensorListInfo, which is already stored, is returned.

2.5 Access Cameras



Description

This state diagram is for surveillance functions that described in the “Access Cameras” use-case. This diagram is for only “Access Cameras” use-case. The use-case is described in the use-case statement.

Each camera has two states, activated or deactivated. And each camera has one password and many functions-zoom, pan, tilt and so on.

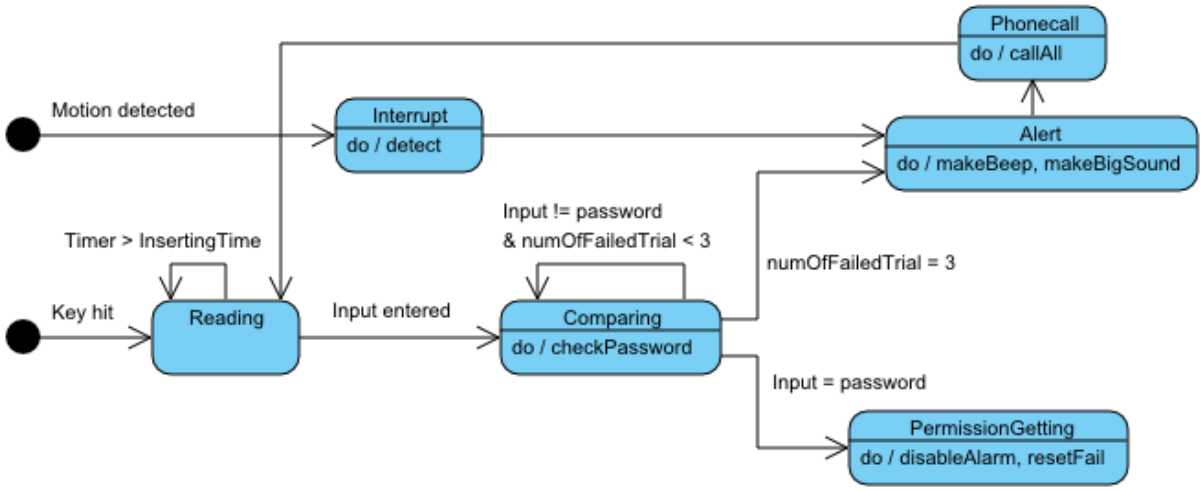
State flow

There is two states in each camera. One is activated, the other is deactivated. When the camera management orders enable(), the camera must turn on. Otherwise the camera must turn off.

In the deactivated or activated state, if user wants to order something functions. Then, the camera management state is loaded. And camera management conducts the order that user wants – Recoding, imaging, controlling and so on.

Camera Management also has initializing or checking function. The function is the search state in the state diagram. When the state is loaded, search the entire camera and indentify the status of camera.

2.6
Alert



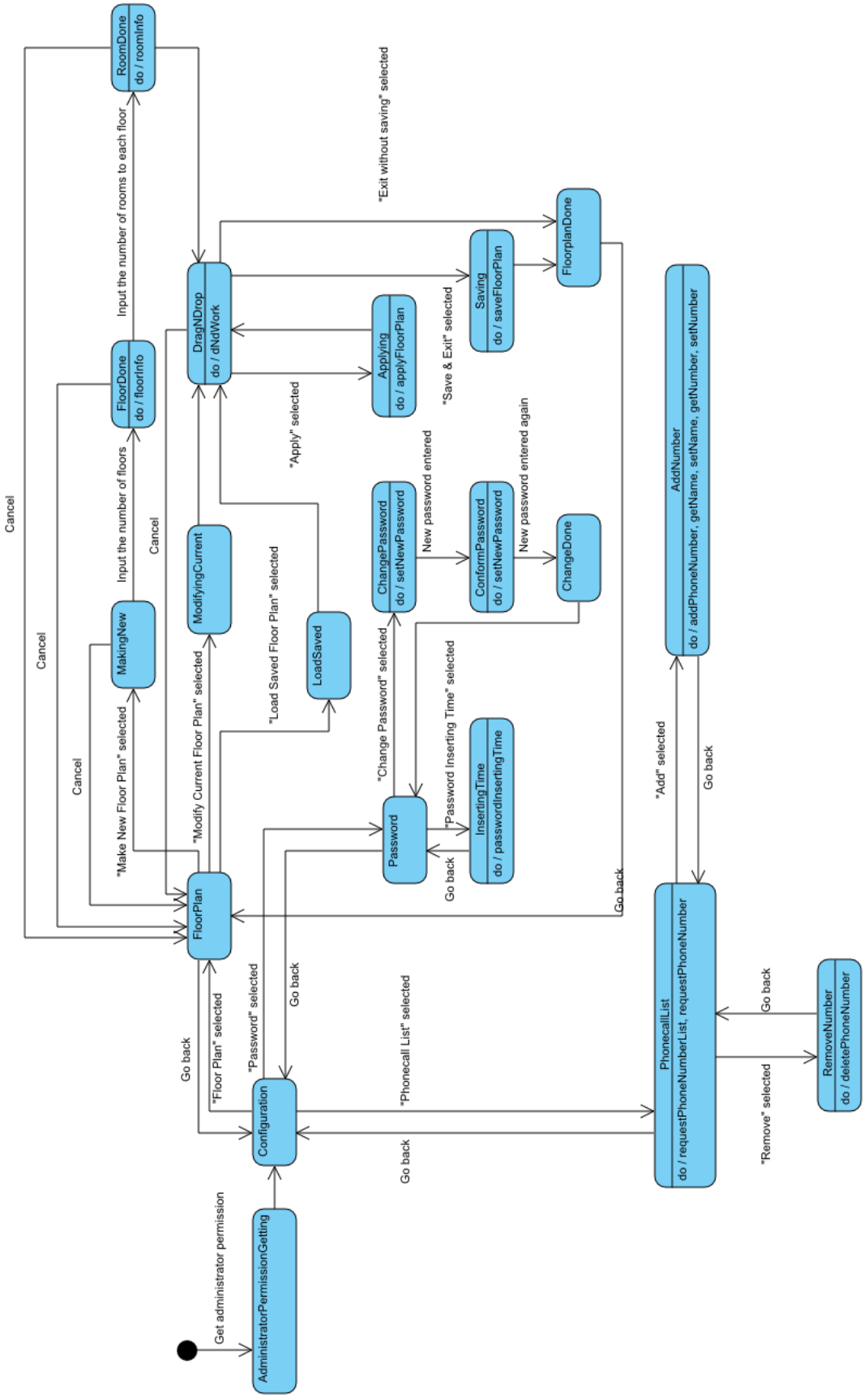
Description

In the emergency situation, which need to sound a warning, SafeHome system warn the urgency of situation using some External Hardware that sounds. Also, when the user goes into the house that security function is activated and input the password, a beep sounds until the password is correct.

State Flow

There are two causes that make Alert performed. One is that user input incorrect password 3 times. The other is that unexpected motion is detected in away activated state. If Alert is performed, alarm sounds and the system calls the numbers in Phonecall list. In both situations, if user inputs correct password, that makes Alert disable then alarming sound will be gone.

2.7 Configuration



Description

In the internet interface, Administrator can modify some information about Safehome, floor-plans, passwords, phonecall lists, and so on. This diagram is based on the use-case statement about configuration. And there are sub-diagrams because each modifying function is too complex to describe one diagram.

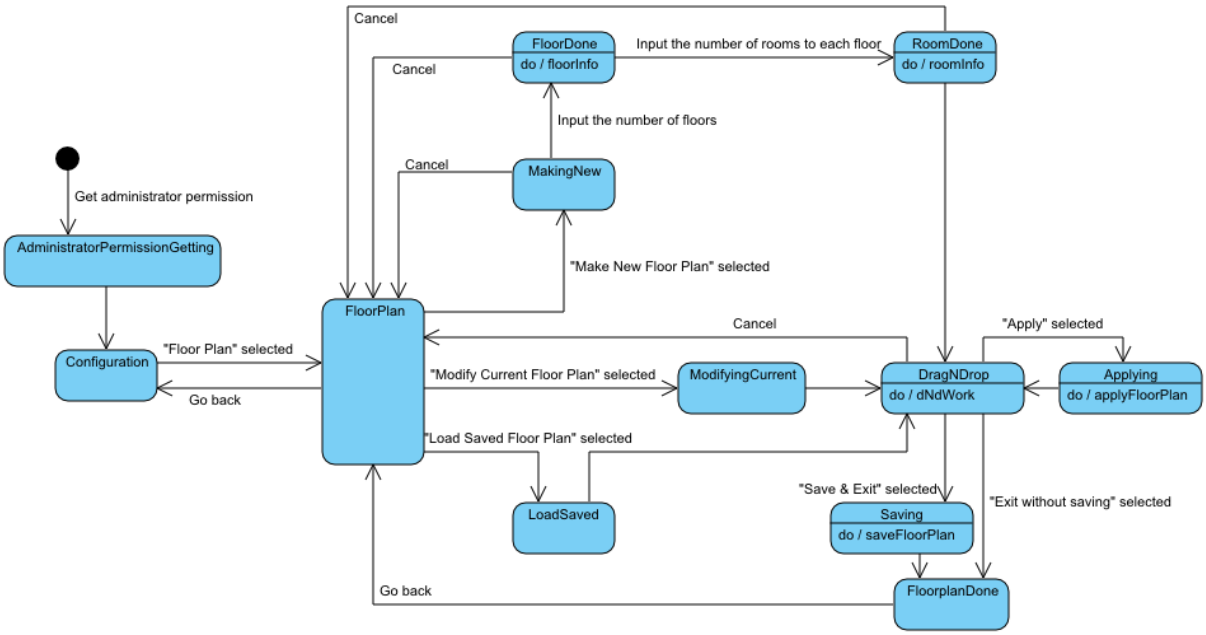
State flow

At first, user selects what user wants to modify. Then the state branches off. And when user wants to go back, the state goes back to the first state.

1) Floor plan
   There are three functions in the modifying floor plan. The state is changed by the user's selection. If user wants to cancel or exit, go back to first state. The detailed description and flow is described in the sub-diagram.
2) Password
   There are two functions in the modifying password. The state is changed by the user's selection. If user wants to cancel or exit, go back to first state. The detailed description and flow is described in the sub-diagram.
3) Phonecall list
   There are three functions in the modifying phonecall list. The state is changed by the user's selection. User can add phone numbers and delete phone numbers. If user wants to cancel or exit, go back to first state. The detailed description and flow is described in the sub-diagram.

Configuration starts with selecting any of those three menus. Only administrator permission is authorized to configure the system. Every state has "Go back" transition to go to previous state, and some states in Floor Plan have "Cancel" transition to go to cancel current state.

2.7.1 Configuration – Floor Plan



State Flow

If administrator chooses “Make new floor plan”,

1. Input the number of floors.
2. Input the number of rooms in each floor.
3. Then the system shows an initial display which has all floors with rooms.
4. Administrator can set doors, windows, cameras, sensors with drag & drop.
5. After finish setting floor plan, administrator can select “Apply” to apply new floor plan or “Save & Exit” to save and then quit or “Exit without saving” to go back.

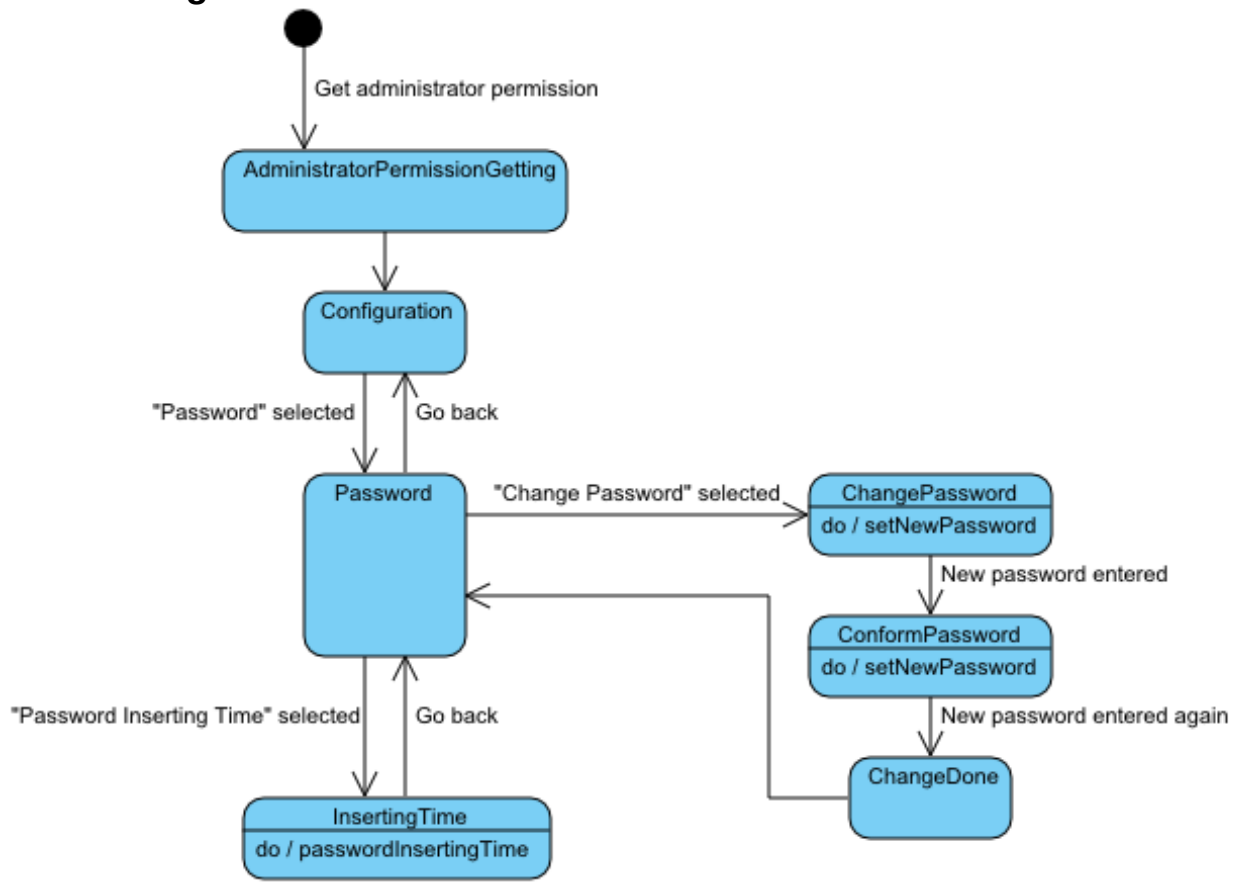
If administrator chooses “Modify current floor plan”

1. The system shows the current floor plan.
2. Administrator can reset doors, windows, cameras, sensors with drag & drop..
3. After finish setting floor plan, administrator can select “Apply” to apply new floor plan or “Save & Exit” to save and then quit or “Exit without saving” to go back.

If administrator chooses “Load saved floor plan”,

1. The system shows the list of the saved floor plan.
2. Administrator selects one of them, and can see selected floor plan.
3. Administrator can reset doors, windows, cameras, sensors with drag & drop.
4. After finish setting floor plan, administrator can select “Apply” to apply new floor plan or “Save & Exit” to save and then quit or “Exit without saving” to go back.

2.7.2 Configuration – Password



State Flow

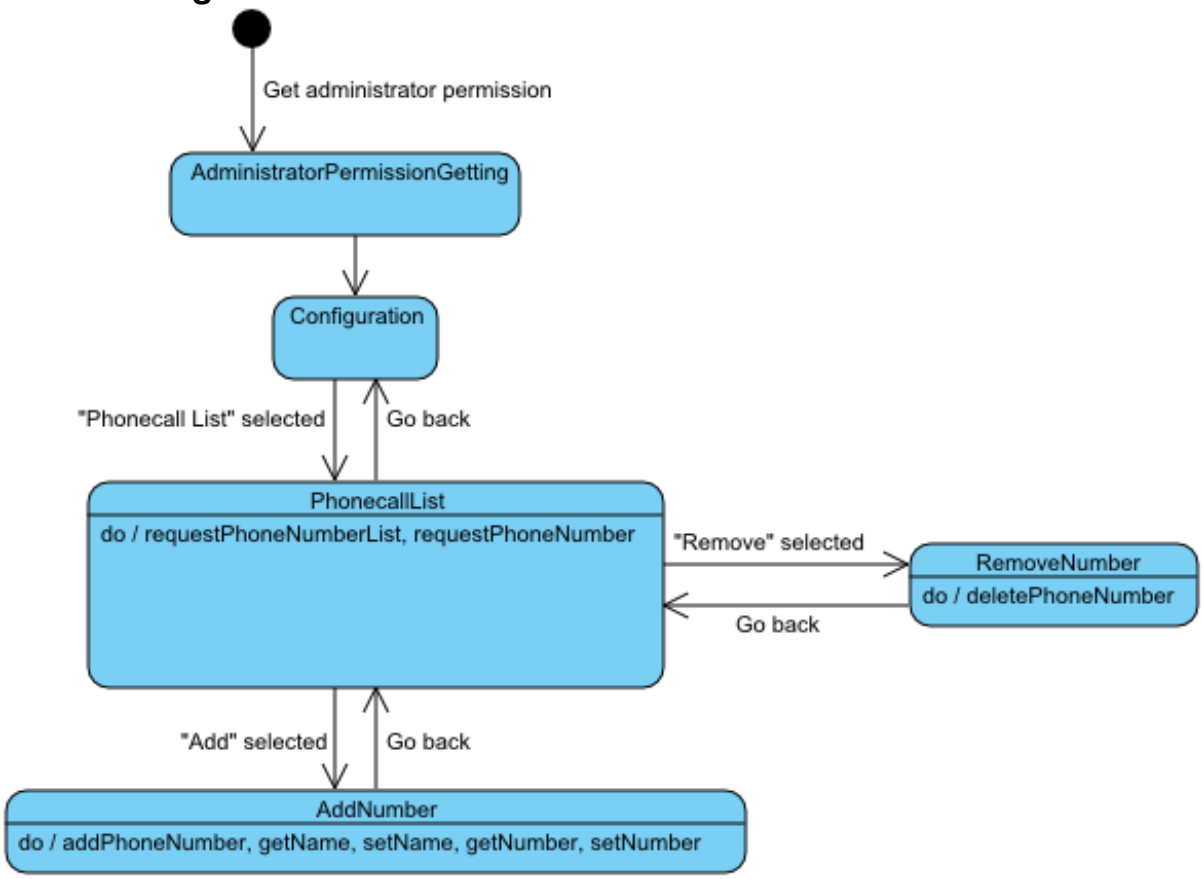
If administrator chooses “Change Password,

1. Administrator inputs new password.
2. Administrator inputs new password again for conforming.
3. Then password is changed and administrator will see previous menu.
4. Administrator can cancel this work in any step with pushing “Cancel” button.

If administrator chooses “Password Inserting Time”,

1. Display shows current password inserting time.
2. Administrator can change the time.
3. Time can be inputted by only numbers, and the unit of time should be in second.
4. Administrator can finish with pushing “Done” button to apply changed setting or “Cancel” to cancel changing.

2.7.3 Configuration – Phonecall list



State Flow

If administrator chooses “Add”, administrator can add a phonecall number to the list.

If administrator chooses “Remove”, administrator can remove a phonecall number from the list.

Appendix A: Team Meeting Reposts

CS350 Software Engineering

Team Meeting Report III-I

Team 4

Who:

Lee Sangyoung,

Kwon Sejoong,

Lee Jieun,

Won Kanghee

When:

2008/04/23 Wed

Where:

Eve room in CS building

Work:

1. Discuss about the previous project.
   - All team members
2. Modify the use-case diagram and make sub use-case diagram
   - All team members
3. Modify the use-case statement.
   - Lee Jieun, Lee Sangyoung.
4. Modify the analysis diagram and make added diagram.
   - Won Kanghee, Kwon Sejoong

Result:

1. Modified use-case diagrams and use-case statements
2. Modified sequence diagrams and activity diagram.

Sign

CS350 Software Engineering

Team Meeting Report III-II

Team 4

Who:

Lee Sangyoung,

Kwon Sejoong,

Lee Jieun,

Won Kanghee

When:

2008/04/24 Thu

Where:

Eve room in CS building

Work:

1. Continue modifying the previous project.
   - All team members
2. Add descriptions to sequence diagrams
   - Won Kanghee, Lee Sangyoung
3. Add traceability to use-case statements
   - Lee Jieun, Kwon Sejoong.
   -

Result:

1. Modified use-case diagrams and use-case statements
2. Modified sequence diagrams and activity diagram.

Sign

CS350 Software Engineering

Team Meeting Report III-III

Team 4

Who:

Lee Sangyoung,

Kwon Sejoong,

Lee Jieun,

Won Kanghee

When:

2008/04/25 Fri

Where:

Eve room in CS building

Work:

1. Continue modifying the previous project.
   - All team members
2. Make the class-diagram
   - All team members

Result:

1. Initial the class-diagram.
2. Modifying the previous project is almost done.

Sign

CS350 Software Engineering

Team Meeting Report III-IV

Team 4

Who:

Lee Sangyoung,

Kwon Sejoong,

Lee Jieun,

Won Kanghee

When:

2008/04/26 Sat

Where:

Eve room in CS building

Work:

1. Make the class-diagram
   - Lee Ji-eun, Kwon Sejoong
2. Make the state-diagram
   - Lee Sangyoung
3. Meeting with Professor Moon.
   - Lee Ji-eun
4. Make initial CRC Cards
   - Won Kanghee
5. Add descriptions to activity diagrams
   - Kwon Sejoong

Result:

1. Class-diagram and initial state-diagram.
2. initial CRC Cards,

Sign

CS350 Software Engineering

Team Meeting Report III-V

Team 4

Who:

Lee Sangyoung,

Kwon Sejoong,

Lee Jieun,

Won Kanghee

When:

2008/04/27 Sun

Where:

Eve room in CS building

Work:

1. Review traceability and modify above all results.
 - Lee Ji-eun, Won Kanghee, Kwon Sejoong.
2. Meeting with Kim Yunho, the TA of SE class
 - Lee Ji-eun, Won Kanghee, Kwon Sejoong.
3. Confirm the class-diagram design.
 - All team members
4. Make state-diagrams.
 - Lee, sang-young.

Result:

1. Class-diagram.
2. Modified use-case diagrams, use-case statements, activity diagrams and sequence diagrams.
3. State diagrams

Sign

CS350 Software Engineering

Team Meeting Report III-VI

Team 4

Who:

Lee Sangyoung,

Kwon Sejoong,

Lee Jieun,

Won Kanghee

When:

2008/04/28 Mon

Where:

Eve room in CS building

Work:

1. Make the documents of this project

- All team members

Result:

1. The final results of the third project/

Sign