

# Project

## Report on Configuration DefaultConfig

### Overridden Properties

#### Subjects:

CG

#### Metaclasses:

CGGeneral

#### Properties:

GeneratedCodeInBrowser: True

# PACKAGES

## Default

GLOBALS:

CLASSES:

---

**ac**

Relations:

**itsThermostat**

Association with Thermostat, Multiplicity of 1, Bi-directional

Operations:

**isFurnaceOn**

Primitive-operation , Public, Return type is OMBoolean

Body

```
return(acOn == TRUE);
```

**startAC**

*this function will turn the ac motor on*

Primitive-operation , Public, Return type is void

Body

```
acOn = TRUE;
```

**stopAC**

*This function will stop the ac*

Primitive-operation , Public, Return type is void

Body

```
acOn = FALSE;
```

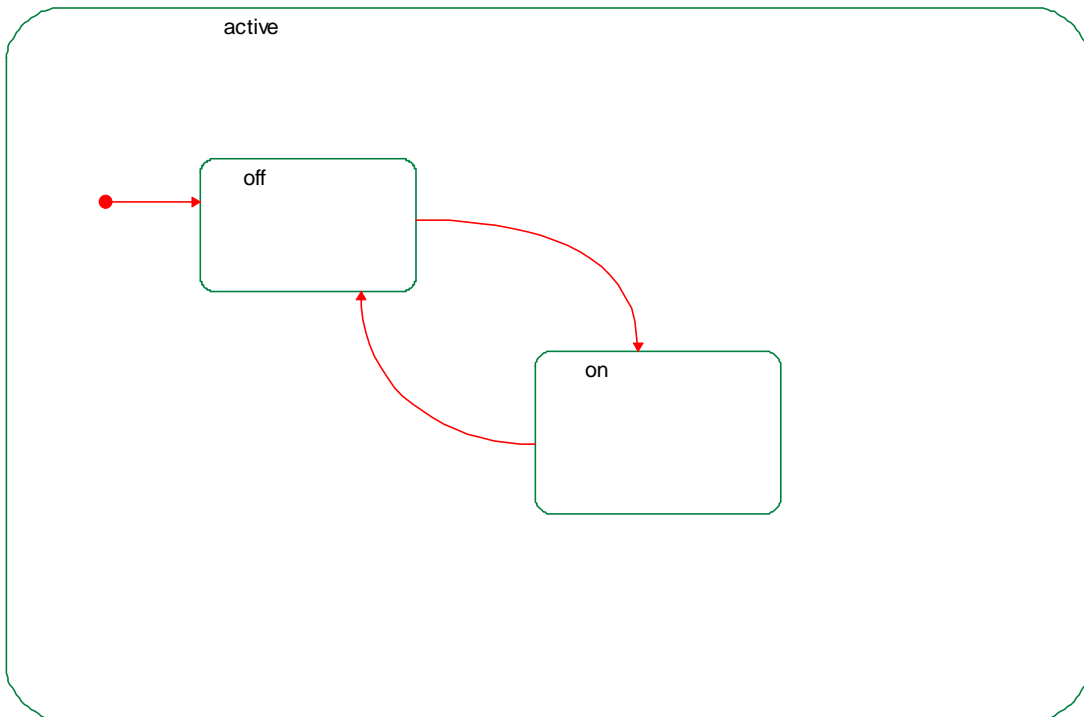
Attributes:

**acOn**

*This attribute will represent if the furnace is on or not.*

Type of OMBoolean, Public

## Statechart



### ROOT

Or-state

Substates:

active

### active

Or-state

Substates:

off

on

Default Transition

Target:

off

### off

Or-state

Out Transition

Target:

on

### on

Or-state

Out Transition

Target:

off

---

### blockSettings

*This class will contain attributes that tell the thermostat how it should act during a specific block of time during the day.*

Operations:

**isIgnoreMaxTemp**

Primitive-operation , Public, Return type is OMBoolean

Body

```
return(ignoreMaxTemp == TRUE);
```

**isIgnoreMinTemp**

Primitive-operation , Public, Return type is OMBoolean

Body

```
return(ignoreMinTemp == TRUE);
```

Attributes:

**blockEnd**

*This attribute will describe what time the current block should end.*

Type of int, Public

**blockStart**

*This time describes when the window should start.*

Type of int, Public

**ignoreMaxTemp**

*This flag will tell the thermostat to ignore the maximum temprature attribute. This can come in handy during the winter when all you are concerned about is maintaining a minimum temperature.*

Type of OMBoolean, Public

**ignoreMinTemp**

*This flag will tell the thermostat to ignore the minimum temperature setting. This can be useful in the summer months when all you are concerned about is keeping the house below the maximum temperature.*

Type of int, Public

**maxTemp**

*This variable will hold the maximum temperature that the thermostat should maintain the house at during the season. The temperature will be represented in degrees Fahrenheit.*

Type of int, Public

**minTemp**

*This varable will hold the minimum temprature that the thermostat should maintain during the season. The temperature will be represented in degrees Fahrenheit.*

Type of int, Public

---

**furnace**

Relations:

**itsThermostat**

Association with Thermostat, Multiplicity of 1, Bi-directional

Operations:

**isFurnaceOn**

Primitive-operation , Public, Return type is OMBoolean

Body

```
return(furnaceOn == TRUE);
```

**startFurnace**

*This operation will start the furnace*

Primitive-operation , Public, Return type is void

Body

```
furnaceOn = TRUE;
```

**stopFurnace**

*This operation will stop the furnace*

Primitive-operation , Public, Return type is void

Body

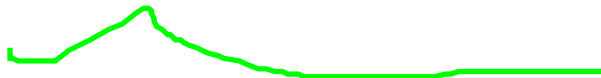
```
furnaceOn = FALSE;
```

Attributes:

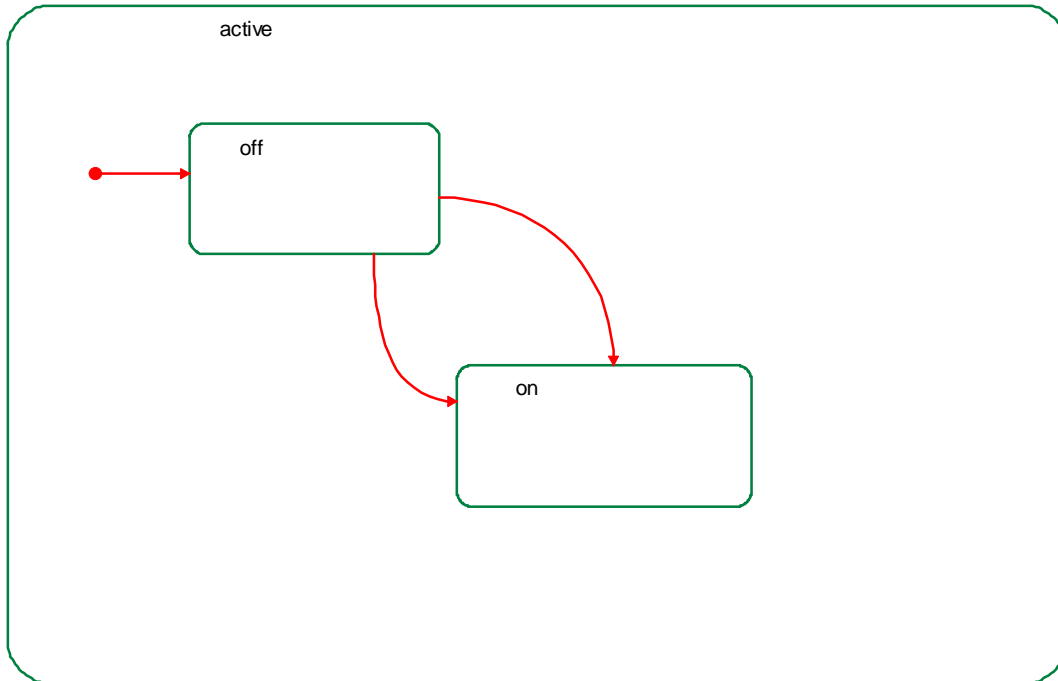
**furnaceOn**

*This represents if the furnace is on, or not.*

Type of OMBoolean, Public



## Statechart



### ROOT

Or-state

Substates:

active

### active

Or-state

Substates:

off

on

Default Transition

Target:

off

### off

Or-state

Out Transition

Target:

on

Out Transition

Target:

on

### on

Or-state

---

## seasonSettings

*This class will hold the settings for each season. So it will hold 4 block objects. You can access and change the settings for each block using accessor functions.*

Relations:

### itsBlockSettings

*This directional association represents the 8 hour blocks in a day that can take up a season. Each season will consist of 4 blocks.*

Association with blockSettings, Multiplicity of 4, Uni-directional

Operations:

**gotoNextBlock**

*This operation will move the current block to the next block*

Primitive-operation , Public, Return type is void

**setCurrentBlock**

*this function will allow the user to set the current block*

Primitive-operation , Public, Return type is void

Args:

int newBlock

*This arg represents the block that we should be running.*

- 1.) morn
- 2.) midday
- 3.) afternoon/evening
- 4.) Nighttime

Attributes:

**currentBlock**

*This is an int that will indicate the block that is currently running.*

- 1.) Early morn block. (until 8 am)
- 2.) midday block
- 3.) afternoon/evening block
- 4.) Night block

Type of int, Public

---

**thermometer**

*This class will be representing a thermometer that's attached to the thermostat. The thermostat will read the current temperature from the thermostat.*

**Overridden Properties**

Subjects:

CG

Metaclasses:

Class

Properties:

Concurrency: active

Relations:

**itsThermostat**

Association with Thermostat, Multiplicity of 1, Bi-directional

Operations:

**decCurrentTemp**

*This function will decrement the current temperature the thermometer is reading.*

Primitive-operation , Public, Return type is void

Body

--currentTemp;

**incCurrentTemp**

*this function will increment the current temperature that the thermometer is reading.*

Primitive-operation , Public, Return type is void

Body

++currentTemp;

Attributes:

**currentTemp**

*This variable will hold the current temperature.*

Type of int, Public

---

**Thermostat**

Relations:

**itsFurnace**

Association with furnace, Multiplicity of 1, Bi-directional

**itsAc**

Association with ac, Multiplicity of 1, Bi-directional

### **itsSeasonSettings**

*This holds SeasonSettings, 4 of them specifically. Each SeasonSetting represents the settings in the thermostat for each season.*

Association with seasonSettings, Multiplicity of 4, Uni-directional

### **itsManualSettings**

*This object will hold the settings defined for the single manual mode for the thermostat.*

Association with blockSettings, Multiplicity of 1, Uni-directional

### **itsThermometer**

Association with thermometer, Multiplicity of 1, Bi-directional

## Operations:

### **changeSeason**

*This function can be called to change the current season of the thermostat. This arg specifies what season to run. Here is a lowdown of the values expected:*

- 1.) Spring
- 2.) Summer
- 3.) Fall
- 4.) Winter

Primitive-operation , Public, Return type is void

### Args:

int seasonType

*This arg specifies what season to run. Here is a lowdown of the values expected:*

- 1.) Spring
- 2.) Summer
- 3.) Fall
- 4.) Winter

## Attributes:

### **aboveMinTemp**

*This variable will tell the system whether to keep the temperature of the house above the Minimum Temperature that's indicated in MinTemp.*

Type of OMBoolean, Public

### **belowMaxTemp**

*This variable will hold a boolean value that tells the system whether to keep the temperature of the house below MaxTemp.*

Type of OMBoolean, Public

### **currentTime**

*This variable holds the current time for the system.*

Type of int, Public

### **inManualMode**

*This boolean flag will indicate if the system is in manual mode.*

Type of OMBoolean, Public

### **MaxTemp**

*This will hold the maximum temperature allowable during the current period. The temperature will be represented in degrees Fahrenheit.*

Type of int, Public

### **MinTemp**

*This will hold the minimum temperature to maintain during the current cycle. The temperature will be represented in degrees Fahrenheit.*

Type of int, Public

### **Temperature**

*This variable holds the temperature that the thermostat is currently reading.*

Type of int, Public

## ACTORS:

---

### AC

*This device is controlled by the thermostat, and is used to cool the house. It also has electronic controls that allow the thermostat to start or stop it.*

Relations:

**itsCool the house**

Association with Cool the house, Multiplicity of 1, Bi-directional

---

### Furnace

*This actor represent the furnace as well as the electronic controls to the furnace. This is what the thermostat talks to when the temperature of the house needs to be increased.*

Relations:

**itsHeat the house**

Association with Heat the house, Multiplicity of 1, Bi-directional

---

### Homeowner

*This actor is the person the Thermostat system is designed to serve. They will be interacting with the system on many levels, including:*

- 1.) Setting up the system
- 2.) Maintaining it.
- 3.) Operating it manually

Relations:

**itsCheck Temperature**

Association with Check Temperature, Multiplicity of 1, Bi-directional

**itsCheck Time**

Association with Check Time, Multiplicity of 1, Bi-directional

**itsManual Mode**

Association with Manual Mode, Multiplicity of 1, Bi-directional

**itsTemperature Control Automation**

Association with Temperature Control Automation, Multiplicity of 1, Bi-directional

**itsEnviornmental Enhancements**

Association with Enviornmental Enhancements, Multiplicity of 1, Bi-directional

---

## USE CASES:

---

### Check Temperature

*This case will display the current temperature on the display for the user. This is also an important case, because it can help the user in a variety of situations:*

- 1.) To figure out what temperature "feels right"
- 2.) Prove to the user the automatic controls are keeping the house at the desired temperature.
- 3.) And to monitor the progress of the thermostat as it changes the temperature of the house - maybe as the house heats up.
- 4.) To show the user the current temperature, so they can enter the Manual mode if the current temperature doesn't meet their needs.

Relations:

**itsHomeowner**

---

### Check Time

*This case occurs when the user wants to see the current time in the display window on the Thermostat.*

*This can also be a very useful feature to the user because:*

- 1.) If the user comes home or wakes up at a certain time, they can look at the thermostat in order to answer the question "by what time do you want the house at this temperature." This is very important when it comes to the enviornmental feature.

2.) Also serves as another useful time telling device around the house. People today are always in a rush, so it's important to have clocks around so they aren't late.

Relations:

**itsHomeowner**

---

### **Cool the house**

This case will talk to and control the air conditioning for the house. The thermostat will use this during either Manual Mode or during the use of automated controls.

Relations:

**itsAC**

---

### **Enviornmental Enhancements**

The main benefit of this feature is that it will take the settings created by the user in the "Temperature Automation Control" case and attempt to modify them in order to improve energy efficiency. This is of a great benefit to the user because it can potentiall reduce monthly energy bills.

This case will only allow the user to enable or disable Smart Enviornmental Enhancements.

Relations:

**itsHomeowner**

---

### **Heat the house**

This case will talk to the furnace telling it to turn on or off, as well as adjust controls of the furnace such as speed.

Relations:

**itsFurnace**

---

### **Manual Mode**

This feature will enable the user to operate the thermostat in a manual mode. This is a very useful feature because:

- 1.) If the weather is unseasonable during a short period, the thermostat might not perform to the users needs.
- 2.) Other special cases such as the user waking up before the pre-scheduled morning warm-up time.

Relations:

**itsHomeowner**

---

### **Temperature Control Automation**

This allows the user to control the automated seasonal settings. The thermostat consists of 4 controllable seasons.

- 1.) Spring
- 2.) Summer
- 3.) Fall
- 4.) Winter

Each of these can control the temerature of the house during certain time blocks in the day, with the day split into 8 hour blocks. During each block the user can tell the system to:

- 1.) Maintain a specific temperature
- 2.) Set a minimum allowable temperature
- 3.) Set a maximum allowable temperature

You can also set a specific start time to heat or cool the house to a temperature (this can be useful in the morning block if you want the heat to turn on at 6:00 because you wake up at 6:30).

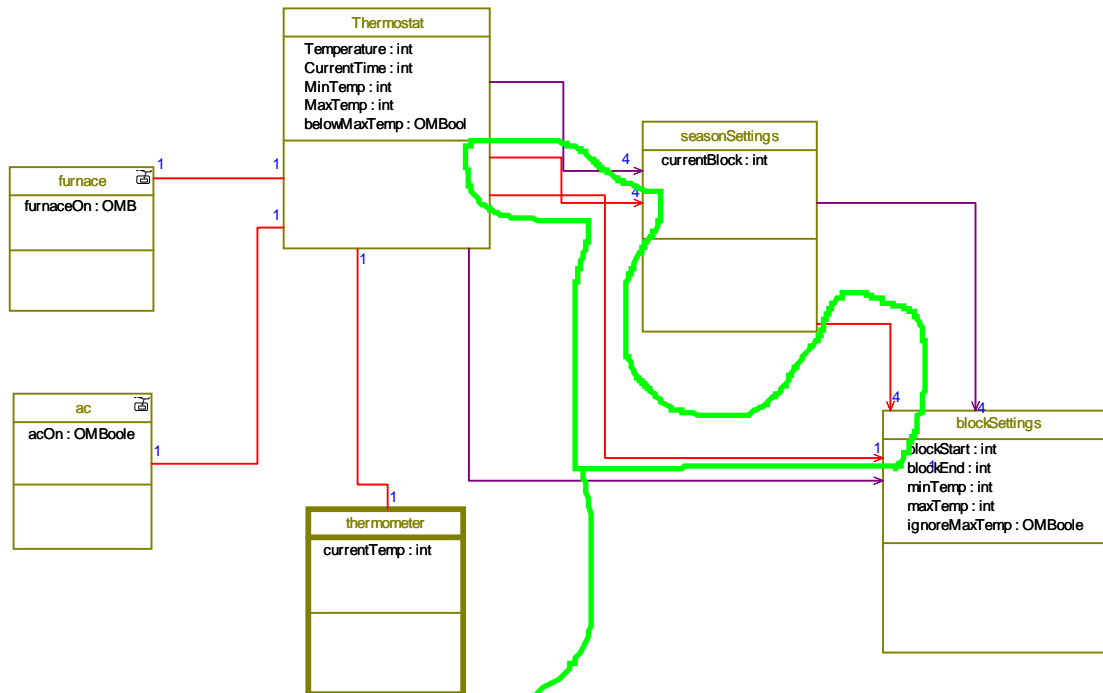
Relations:

**itsHomeowner**

# OBJECT MODEL DIAGRAMS

## Thermostat

This is the Object Model Diagram of my Thermostat system.



You don't need both of these relationships.

# USE CASE DIAGRAMS

## Thermostat

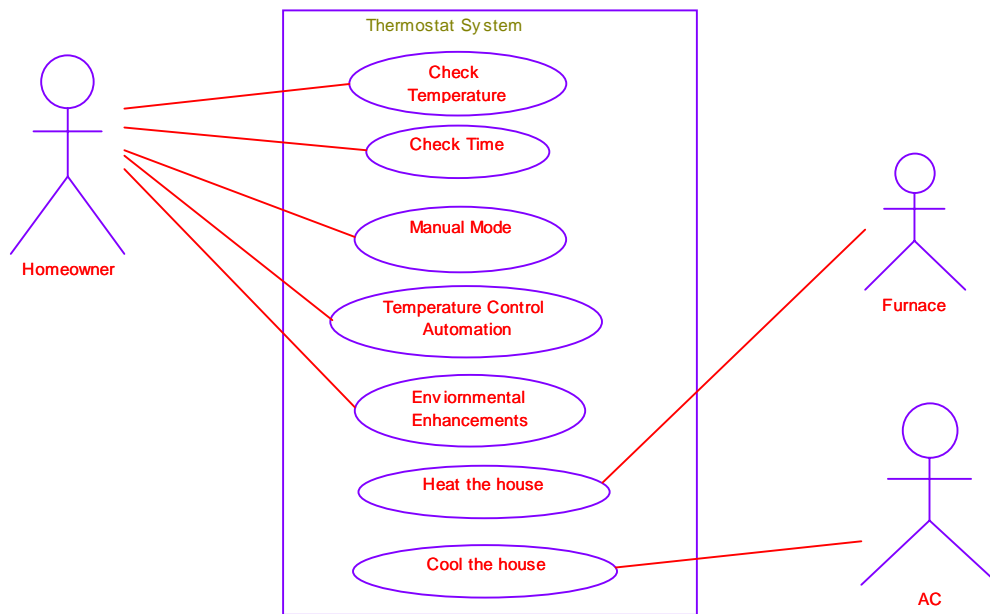
*This use case represents the capability of the Thermostat. This includes 3 actors, and 7 specific use cases.*

*Responsibilities: The thermostat is responsible for controlling the heating and air conditioning of a house, and maintaining its internal climate.*

*Actors:*

- 1.) Homeowner and/or resident - represents the main user of this system.*
- 2.) Furnace - This represents the furnace that's used to heat the house.*
- 3.) A/C - The external system that is used to cool the house.*

*This use case essentially provides a visual picture of the "black box" functionality of the thermostat system.*



Very well done so far. Keep up the good work.  
Neil

# COMPONENTS

## DefaultComponent

### COMPONENT SETTINGS:

Build type: Executable

### CONFIGURATIONS:

---

#### **DefaultConfig**

Scope type: Explicit

Instrumentation type: None

Time-model type: Real-time

Statechart generation type: Flat

### FILES AND FOLDERS: