**Functionality Outline**

Sensor Simulation

Programming Assignment Two

July 28, 2016

Prepared By:

**Jared Nixon**

Prepared For:

**Dr. Rick Coleman**  
CS 307, Object Oriented Programming

Class Outline

Jared Nixon

**Table of Contents**

1. Table of Contents . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2
2. System Overview . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3
3. Main . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3
4. Class Simulation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5
5. Class SensorMount. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6
6. Class SensorFactory . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7
7. Class Sensor . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7
8. Class AirSensor . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8
9. ClassEarthSensor . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8
10. Class WaterSensor . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8
11. Class AirFactory . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9
12. Class EarthFactory . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9
13. Class WaterFactor . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9
14. Class DisplayFactory . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10
15. Class Display . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 11
16. Class CRT . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 11
17. Class Plotter . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 12
18. Class Meter . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 13
19. System Overview

The functionality of this system will simulate sensors reading information from the environment and relaying the readings back to the user. The user will also be allowed to subscribe and unsubscribe sensors via pressing the ‘s’ key. This simulation will be generated from an XML file, which will contain all information about the display devices and sensors, including how many of each and their respective types. The class outline below will help illustrate the functionality of this system and list some of the intended responsibilities of each designed object.

1. Object Functionality
   1. File: Main.cpp
      1. **Main()**

Instantiate the Simulation object.

Call Simulation::initializeSimulation.

Call Simulation::runSimulation.

* 1. File: Simulation.cpp
     1. **Simulation()**

Get an instance for the airFactory object.

Get an instance for the earthFactory object.

Get an instance for the waterFactory object.

Get an instance for the displayFactory object.

Instantiate the sensorMount object.

Instantiate a tempSensor object.

Instantiate a tempDisplay object.

* + 1. **~Simulation()**

Delete the sensorMount object.

Delete all sensor objects.

Delete all Factory objects.

* + 1. **initializeSimulation()**

Request input file name from the user.

Create an instance of the data parser and set it equal to the getInstance function.

Gather all air sensors until there are no more air sensors to be read.

*This will be done in a while loop.*

*The temporary sensor will also be used to store temporary data.*

*Pass the tempSensor object to airFactory, and set the return equal to an airSensor.*

*Push the sensor onto the sensorContainer vector.*

Gather all earth sensors until there are no more earth sensors to be read.

*This will be done in a while loop.*

*The temporary sensor will also be used to store temporary data.*

*Pass the tempSensor object to earthFactory, and set the return equal to an earthSensor.*

*Push the sensor onto the sensorContainer vector.*

Gather all water sensors until there are no more water sensors to be read.

*This will be done in a while loop.*

*The temporary sensor will also be used to store temporary data.*

*Pass the tempSensor object to waterFactory, and set the return equal to an waterSensor.*

*Push the sensor onto the sensorContainer vector.*

Gather data display by display from the file until all have been read.

*This will be done in a while loop.*

*The temporary display will also be used to store temporary data.*

*Pass the tempDisplay to the displayFactory, and push the return value into the vector.*

Output current sensorMount configuration (sensors and display devices)

* + 1. **runSimulation()**

Start the timer loop.

Check if it is time to update the sensor data.

*This will happen every five seconds.*

If it is time to update, then call SensorMount->propogateData.

Increment the next output time for the next five seconds.

See if a key has been pressed.

If a key has been pressed, get the pressed key.

Compare the pressed key to the key ‘s’

If that was the pressed key, update a menu to the user.

For displayContainer size

For Display CountID[i]

If the toggle bool is true then it is monitoring.

If the toggle bool is false then it is not monitoring.

Watch for a cin with the structure [display sensor] to see which to turn off. EX: 1 220

For Display CountID @ the location indicated by the user [display – 1]

If the toggle is true at that point && the ID == the sensor input by the user

Set toggle to false.

Else if the toggle is false at that point && the ID == the sensor input by the user

Set toggle to true.

* + 1. **getInstance()**

Set up a static Simulation object and set it equal to NULL.

If the object == NULL

Call the private constructor.

Return the instance of the Simulation.

* 1. File: SensorMount.cpp

**2.3.1 SensorMount()**

No specific purpose defined for Assignment One.

**2.3.2 ~SensorMount()**

No specific purpose defined for Assignment One.

**2.3.3 propogateData(vector<Sensor>sensorContainer, vector<Display>displayContainer)**

For each display device in the displayContainer

For each sensor in the sensorContainer.

If getDisplayIDCount > 1

If getDisplayID[k] == getSensorID[k]

If the toggle is set to true for the displayContainer[i] object

Call displayContainer[i]->Update and pass it the sensor object.

Else

Output that the sensor is not being monitored (so the user knows a sensor is still there)

**2.3.4 getInstance()**

Set up a static SensorMount object and set it equal to NULL.

If the object == NULL

Call the private constructor.

Return the instance of the SensorMount object.

* 1. File: Sensor.cpp

**2.4.1 Sensor()**

Call the srand funcftion to seed the random number generator.

**2.4.2 ~Sensor()**

No specific purpose defined for Assignment One.

**2.4.3 GenerateData(double min, double max)**

Return a random integer between the min parameter and the max parameter using the rand function.

**2.4.4 setType(char\* type)**

Use strcpy to copy the string in the char array parameter into the char array m\_chType private variable.

**2.4.5 setMaterial(char\* material)**

Use strcpy to copy the string in the char array parameter into the char array m\_chMaterial private variable.

**2.4.6 setSensorID(int\* id)**

Set the private variable m\_iSensorID equal to the dereferenced pointer of the int array parameter.

**2.4.7 setUnits(char\* units)**

Use strcpy to copy the string in the char array parameter into the char array m\_chUnits private variable.

**2.4.8 setValueMax(double max)**

Set the double parameter equal to the private variable m\_dValueMax.

**2.4.9 setValueMin(double min)**

Set the double parameter equal to the private variable m\_dValueMin.

**2.4.10 getType()**

Return the type of sensor associated with the tempSensor.

**2.4.11 getMaterial()**

Return the type of material associated with the tempSensor.

**2.4.12 getSensorID()**

Return the ID associated with the tempSensor.

**2.4.13 getUnits()**

Return the units that the tempSensor is being read in.

**2.4.14 getValueMax()**

Return the maximum data value the tempSensor can generate.

**2.4.15 getValueMin()**

Return the minimum data value the tempSensor can generate.

* 1. File: SensorFactory.cpp

**2.5.1 SensorFactory()**

Not specific purpose defined for Assignment Two.

This is simply an interface for the Air, Earth, and Water factories.

**2.5.2 ~SensorFactory()**

Not specific purpose defined for Assignment Two.

* 1. File: AirSensor.cpp

**2.6.1 AirSensor(Sensor\* tempSensor)**

Take the tempSensor->getType() return and use it as the parameter in setType() : function inheritance.

Take the tempSensor->getMaterial() return and use it as the parameter in setType() : function inheritance.

Take the tempSensor->getSensorID() return and use it as the parameter in setType() : function inheritance.

Take the tempSensor->getUnits() return and use it as the parameter in setType() : function inheritance.

Take the \*tempSensor->getValueMax() return and use it as the parameter in setType() : function inheritance.

Take the \*tempSensor->getValueMin() return and use it as the parameter in setType() : function inheritance.

**2.6.2 ~AirSensor()**

No specific purpose defined for Assignment Two.

* 1. File: EarthSensor.cpp

**2.7.1 EarthSensor(Sensor\* tempSensor)**

Take the tempSensor->getType() return and use it as the parameter in setType() : function inheritance.

Take the tempSensor->getMaterial() return and use it as the parameter in setType() : function inheritance.

Take the tempSensor->getSensorID() return and use it as the parameter in setType() : function inheritance.

Take the tempSensor->getUnits() return and use it as the parameter in setType() : function inheritance.

Take the \*tempSensor->getValueMax() return and use it as the parameter in setType() : function inheritance.

Take the \*tempSensor->getValueMin() return and use it as the parameter in setType() : function inheritance.

**2.7.2 ~EarthSensor()**

No specific purpose defined for Assignment Two.

* 1. File: WaterSensor.cpp

**2.8.1 WaterSensor (Sensor\* tempSensor)**

Take the tempSensor->getType() return and use it as the parameter in setType() : function inheritance.

Take the tempSensor->getMaterial() return and use it as the parameter in setType() : function inheritance.

Take the tempSensor->getSensorID() return and use it as the parameter in setType() : function inheritance.

Take the tempSensor->getUnits() return and use it as the parameter in setType() : function inheritance.

Take the \*tempSensor->getValueMax() return and use it as the parameter in setType() : function inheritance.

Take the \*tempSensor->getValueMin() return and use it as the parameter in setType() : function inheritance.

**2.8.2 ~WaterSensor()**

No specific purpose defined for Assignment Two.

* 1. File: AirFactory.cpp

**2.9.1 AirFactory(Sensor\* tempSensor)**

No specific purpose defined for Assignment Two.

**2.9.2 ~AirFactory()**

No specific purpose defined for Assignment Two.

**2.9.3 getInstance()**

Set up a static AirFactory object and set it equal to NULL.

If the object == NULL

Call the private constructor.

Return the instance of the AirFactory object.

**2.9.4 CreateSensor(Sensor\* tempSensor)**

Instantiate an airSensor object by passing the tempSensor to the airSensor constructor.

Return the airSensor

* 1. File: EarthFactory.cpp
     1. **EarthFactory(Sensor\* tempSensor)**

**2.10.2 ~EarthFactory()**

No specific purpose defined for Assignment Two.

**2.10.3 getInstance()**

Set up a static AirFactory object and set it equal to NULL.

If the object == NULL

Call the private constructor.

Return the instance of the AirFactory object.

**2.10.4 CreateSensor(Sensor\* tempSensor)**

Instantiate an earthSensor object by passing the tempSensor to the earthSensor constructor.

Return the earthSensor

2.11 File WaterFactory.cpp

* + 1. **WaterFactory(Sensor\* tempSensor)**.

**2.11.2 ~WaterFactory()**

No specific purpose defined for Assignment Two.

**2.11.3 getInstance()**

Set up a static AirFactory object and set it equal to NULL.

If the object == NULL

Call the private constructor.

Return the instance of the AirFactory object.

**2.11.4 CreateSensor(Sensor\* tempSensor)**

Instantiate an waterSensor object by passing the tempSensor to the waterSensor constructor.

Return the waterSensor

2.12 File DisplayFactory.cpp

**2.12.1 DisplayFactory()**.

No specific purpose defined for Assignment Two.

**2.12.2 ~DisplayFactory()**

No specific purpose defined for Assignment Two.

**2.12.3 getInstance()**

Set up a static DisplayFactory object and set it equal to NULL.

If the object == NULL

Call the private constructor.

Return the instance of the DisplayFactory object.

**2.12.4 CreateDisplay(Sensor\* tempDisplay)**

Compare the string of the DisplayType to “CRT”

If that string compare == 0

Instantiate a crtDisplay by calling its constructor and passing it the tempDisplay

Return the crtDisplay.

Compare the string of the DisplayType to “METER”

If that string compare == 0

Instantiate a meterDisplay by calling its constructor and passing it the tempDisplay

Return the meterDisplay

Compare the string of the DisplayType to “PLOTTER”

If that string compare == 0

Instantiate a plotterDisplay by calling its constructor and passing it the tempDisplay

Return the plotterDisplay

2.13 File: Display.cpp

**2.13.1 Display()**

No specific purpose defined for Assignment Two.

**2.13.2 ~Display**

No specific purpose defined for Assignment Two.

* + 1. **Update(Display\* display, Sensor\* sensor, int Kindex, int data)**

Implemented in the child classes.

* + 1. **setDisplayType(char\* type)**

Use strcpy to copy the string in the char array parameter into the char array m\_chType private variable.

**2.13.5 setDisplayID(int\* id)**

For the number of displayCountID’s associated with the display device, set m\_iID[i] = id[i] to copy thecontents of the parameter int array to the m\_iID private int array.

**2.13.6 setDisplayCountID(int\* countID)**

Set the int array parameter equal to the m\_iCountID private variable.

**2.13.7 getDisplayType()**

Return the type of display associated with the tempDisplay.

**2.13.8 getDisplayID()**

Return the type of ID associated with the tempDisplay.

* + 1. **getDisplayCountID()**

Return the display count associated with the tempDisplay.

* + 1. **InitStruct()**

For the number of Sensor IDs associated with the Display.

Set the structure variable for that toggle bool to true. (Monitoring by default)

Set the structure variable for the ID equal to the ID (so each sensor associated with the display has a structure to itself. This is necessary so when you unmonitor a sensor, if there are two of the same ID, it will not unmonitor them both.)

* + 1. **getStructure()**  
       Return the structure associated with the sensor of the display.

2.14 File: CRT.cpp

**2.14.1 CRT()**

Take the tempDisplay->getDisplayType() return and use it as the parameter in setDisplayType() : function inheritance.

Take the tempDisplay->getDisplayCountID() return and use it as the parameter in setDisplayCountID() : function inheritance.

Take the tempDisplay->getDisplayID() return and use it as the parameter in setDisplayID() : function inheritance.

Function call to initialize the structure for all sensors being read by each display device.

* + 1. **~CRT()**

No specific purpose defined for Assignment Two.

* + 1. **Update(Sensor\* sensor)**

Output the sensor information to the user (sensor ID, Type, Material, Units)

Generate Data based on the max and min value (Data is no longer just PUSHED to the display device, it is now generated when sensor mount tells the observe, the device, that it is time to update and it gets the information it needs.)

2.15 File: Plotter.cpp

**2.15.1 Plotter()**

Take the tempDisplay->getDisplayType() return and use it as the parameter in setDisplayType() : function inheritance.

Take the tempDisplay->getDisplayCountID() return and use it as the parameter in setDisplayCountID() : function inheritance.

Take the tempDisplay->getDisplayID() return and use it as the parameter in setDisplayID() : function inheritance.

Function call to initialize the structure for all sensors being read by each display device.

**2.15.2 ~Plotter()**

No specific purpose defined for Assignment Two.

* + 1. **Update(Sensor\* sensor)**

Output the sensor information to the user (sensor ID, Type, Material, Units)

Generate Data based on the max and min value (Data is no longer just PUSHED to the display device, it is now generated when sensor mount tells the observe, the device, that it is time to update and it gets the information it needs.)

* 1. File: Meter.cpp

**2.16.1 Meter()**

Take the tempDisplay->getDisplayType() return and use it as the parameter in setDisplayType() : function inheritance.

Take the tempDisplay->getDisplayCountID() return and use it as the parameter in setDisplayCountID() : function inheritance.

Take the tempDisplay->getDisplayID() return and use it as the parameter in setDisplayID() : function inheritance.

Function call to initialize the structure for all sensors being read by each display device.

**2.16.2 ~Meter()**

No specific purpose defined for Assignment Two.

* + 1. **Update(Sensor\* sensor)**

Output the sensor information to the user (sensor ID, Type, Material, Units)

Generate Data based on the max and min value (Data is no longer just PUSHED to the display device, it is now generated when sensor mount tells the observe, the device, that it is time to update and it gets the information it needs.)