**Functionality Outline**

Sensor Simulation

Programming Assignment One

July 5, 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 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3
5. Class SensorMount. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4
6. Class Sensor . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5
7. Class Cadmium . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6
8. Class CarbonMonoxide . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6
9. Class Chlorine . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7
10. Class NuclearWaste. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7
11. Class OilResidue . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8
12. Class RawSewage . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8
13. Class Display . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8
14. Class CRT . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9
15. Class Plotter . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10
16. Class Meter . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10
17. System Overview
    1. The functionality of this system will simulate sensors reading information from the environment and relaying the readings back to the user. 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.
18. Object Functionality
    1. File: Main.cpp
       1. **Main()**

Instantiate the Simulation object.

Call Simulation::initializeSimulation.

Call Simulation::runSimulation.

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

Instantiate the sensorMount object.

Instantiate a tempSensor object.

Instantiate a tempDisplay object.

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

Delete the sensorMount object.

Delete all specific sensor objects.

* + 1. **initializeSimulation()**

Request input file name from the user.

Create an instance of the data parser.

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

*This will be done in a while loop.*

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

Instantiate specific sensors with respect to the data parser info.

Store sensors in 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.*

Instantiate specific display devices with respect to the data parser info.

Store display devices in the displayContainer (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.

* 1. File: SensorMount.cpp

**2.2.1 SensorMount()**

No specific purpose defined for Assignment One.

**2.2.2 ~SensorMount()**

No specific purpose defined for Assignment One.

**2.2.3 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]

Generate a random number

Call displayContainer::Update and pass in the display, sensor, k, and data.

Else if getDisplayID == getSensorID (Because DisplayIDCount == 1)

Set k equal to 0 for the only DisplayID in the integer array.

Generate a random number

Call displayContainer::Update and pass in the display, sensor, k, and data.

* 1. File: Sensor.cpp

**2.2.1 Sensor()**

Call the srand funcftion to seed the random number generator.

**2.2.2 ~Sensor()**

No specific purpose defined for Assignment One.

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

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

**2.2.4 setType(char\* type)**

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

**2.2.5 setMaterial(char\* material)**

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

**2.2.6 setSensorID(int\* id)**

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

**2.2.7 setUnits(char\* units)**

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

**2.2.8 setValueMax(double max)**

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

**2.2.9 setValueMin(double min)**

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

**2.2.10 getType()**

Return the type of sensor associated with the tempSensor.

**2.2.11 getMaterial()**

Return the type of material associated with the tempSensor.

**2.2.12 getSensorID()**

Return the ID associated with the tempSensor.

**2.2.13 getUnits()**

Return the units that the tempSensor is being read in.

**2.2.14 getValueMax()**

Return the maximum data value the tempSensor can generate.

**2.2.15 getValueMin()**

Return the minimum data value the tempSensor can generate.

* 1. File: Cadmium.cpp

**2.2.1 Cadmium(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.

**2.2.2 ~Cadmium()**

No specific purpose defined for Assignment One.

* 1. File: CarbonMonoxide.cpp

**2.2.1 CarbonMonoxide(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.

**2.2.2 ~CarbonMonoxide()**

No specific purpose defined for Assignment One.

* 1. File: Chlorine.cpp

**2.2.1 Chlorine(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.

**2.2.2 ~Chlorine()**

No specific purpose defined for Assignment One.

* 1. File: NuclearWaste.cpp

**2.2.1 NuclearWaste(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.

**2.2.2 ~NuclearWaste()**

No specific purpose defined for Assignment One.

* 1. File: OilResidue.cpp

**2.2.1 OilResidue(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.

* + 1. **~OilResidue()**

No specific purpose defined for Assignment One.

2.10 File: RawSewage.cpp

**2.2.1 RawSewage(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.

* + 1. **~RawSewage()**

No specific purpose defined for Assignment One.

2.11 File: Display.cpp

**2.2.1 Display()**

No specific purpose defined for Assignment One.

**2.2.2 ~Display**

No specific purpose defined for Assignment One.

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

Note: At this point, sensorID and deviceID match up and are ready to be sent to the output for the user.

Output the display device type.

Output the display device ID.

Output the sensor type and material.

Output the current sensor reading and the units read in.

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

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

**2.2.5 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.

**2.2.6 setDisplayCountID(int\* countID)**

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

**2.2.7 getDisplayType()**

Return the type of display associated with the tempDisplay.

**2.2.8 getDisplayID()**

Return the type of ID associated with the tempDisplay.

* + 1. **getDisplayCountID()**

Return the display count associated with the tempDisplay.

2.12 File: CRT.cpp

**2.2.1 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.

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

No specific purpose defined for Assignment One.

2.13 File: Plotter.cpp

**2.2.1 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.

**2.2.2 ~Plotter()**

No specific purpose defined for Assignment One.

* 1. File: Meter.cpp

**2.2.1 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.

**2.2.2 ~Meter()**

No specific purpose defined for Assignment One.