Functionality Outline

for

Oil Field Monitoring

Programming Assignment 1
June 30, 2015

Prepared by
Timothy Finnegan
tbf0005@uah.edu

Prepared for:
Dr. Rick Coleman
CS 307, Object Oriented Programming in C++
Computer Science Department
University of Alabama in Huntsville

Table of Contents

1.0 System Overview

Oilfield Instrumentation-USA relies on a complex array of sensors on each drilling rig to monitor the state of all operations. These sensors feed data to a server mounted on the rig. The server is connected to a network which enables workers on there rig as well as managers back at the company offices to monitor all aspects of the operations. This program has been designed to simulate those sensors on oil rigs and display the relevant information in a command prompt.

2.0 Relevant Terms and Acronyms

- ROP - an acronym for rate of penetration, the rate at which the drill bit is drilling
- PSI - pounds per square inch, a unit of measurement for pressure
- BBL - an abbreviation for barrels (of oil in this instance)
- casing pressure - pressure inside of the bit casing

3.0 Object Functionality

3.1 SimulationMain.cpp

main()

Instatiate Simulation object

Call Simulation::initSim()

Call Simulation::startServer()

3.2 Simulation.cpp

initSim()

Create instances of Server

startServer()

Create instances of data parser

Query user for name of the data file

```
Call Server::run()
Get data to define each simulation object from data parser
        while(getnewline returns values)
                getWellCount()
                getWellData()
                getSensorData()
Create instances of Well, Sensor, and Display objects
        Well *w = new Well()
        Sensor *s = new Sensor()
        Display *d = new Sensor()
```

3.3 Server.cpp

```
run()
        Begin timer loop
                Check current time
                        if time to update
                                request data from all simulation objects
                                        for each well instance
                                                call Well::getData()
                                call Display::output()
                                update time for next report
                        check for keyboard input
                        if “keyboard input exists”
                                handle input
                                        if input is W
                                                if input is A
                                                        Call Well::subscribe(Well *w)
                                                if input is R
                                                        Call Well::unsubscribe(Well *w)
                                        if input is S
                                                switch (choose from vector of subcribers)
                                                        if input is A
                                                                Call Sensor::subscribe(Sensor *s)
```

```
                                        if input is R
                                            Call Sensor::unsubscribe(Sensor *s)
                                    if input is Q
                                        exit simulation
        End timer loop
```

3.4 Display.cpp

```
output()
        collect updated data from server
        cout each piece of sensor data for each well

unsubscribe(Well *w)
        for(Well vector iterator = wells.begin(); iterator != wells.end(); iterator++)
            if (w = *iterator)
                erase Well instance
                return true
            return false

subscribe(Well *w)
        push_back an instance of Wells into a vector of Well instances
```

3.5 Well.cpp

```
getWell()
        create Well instance
        return Well instance

getData()
        call getters for each sensor for the Well being requested
```

3.6 Sensor.cpp

```
report()
        sensor subclasses will implement this
```

3.7 HoleDepth.cpp

getHoleDepth()

    return m_holeDepth

setHoleDepth()

    set m_holeDepth to rand()

3.8 ROP.cpp

getROP()

    return m_ROP

setROP()

    set m_ROP to rand()

3.9 BitDepth.cpp

getBitDepth()

    return m_bitDepth

setBitDepth()

    set m_bitDepth to rand()

3.10 PumpPressure.cpp

getPumpPressure()

    return m_pumpPressure

setPumpPressure()

    set m_pumpPressure to rand()

3.11 CasingPressure.cpp

getCasingPressure()

return m_casingPressure

setCasingPressure()

set m_casingPressure to rand()

3.12 TorqueMax.cpp

getTorqueMax()

return m_torqueMax

setTorqueMax()

set m_torqueMax to rand()

3.13 FlowOut.cpp

getFlowOut()

return m_flowOut

setFlowOut()

set m_flowOut to rand()

3.14 MudPitVolume.cpp

getMudPitVolume()

return m_mudPitVolume

setMudPitVolume()

set m_mudPitVoume to rand()