Garden Sprinkler Project

|  |  |
| --- | --- |
| Document Version | 0.01 |
| Document Status | Initial Draft |
| Issue Date | 2013-03-1 |
| Project | 1 |
| Term | Winter 2013 |
| Course | Object Oriented Analysis And Design |
| Author | Devi Kandasamy |

**Abstract:**

In this project, I have implemented a 3-tiered architecture consisting of Java Swing-based UI, business logic code in Java and a backend file store that consists of files and file-accessor classes. The GUI provides interfaces for users to configure temperature-based daily sprinkler schedules, monitor status of sprinklers and sprinkler zones and determine total water consumption of sprinklers. The business-logic component has standard interfaces with GUI and the file store components to be able to process sprinkler schedule and status changes to dynamically reflect changes in water consumption and sprinkler status on the Sprinkler map.

**Table of Contents**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

1. Phases of project 2
2. Learning Outcomes
3. Description of Code 3
4. Storage Location
5. Software Used
6. Code Overview
7. How to execute code?
8. How to parse input?
9. Challenges
10. JUnit
11. Screenshots

# **Phases of project**

The phases followed for the application development are

Phase 1: Requirements gathering

The below are the requirements implemented for the garden sprinkler system

* + Program the system (with a weekly schedule) to start the sprinklers at designated times
  + Enable and disable the system
  + Show the activation of the sprinklers
  + Display the status of the sprinklers
  + Display the total water usage
  + Program the system to activate/deactivate the sprinklers based on temperatures.
  + Adjust the temperatures to show the activation/deactivation of the sprinklers.
  + Program the system to activate/deactivate the sprinkler groups (and individual sprinklers)
  + Display a map/schematic of the garden layout and the location of sprinklers
  + Display a graph showing the water usage by the sprinkler groups

Phase 2: Analysis

1. Class Responsiblity Collaboration(CRC) cards are designed. CRC model gathers the class resposibility and its collaborator classes.

2. UML diagram for each class to define the interaction between a role and the system

Phase 3: Design

1. Class diagram
2. Sequence diagrams
3. State diagram
4. UML diagrams

Phase 4: Implementation

Explained in detail on section 2.0

Phase 5: Testing

Explained in detail on section

# **Learning Outcomes**

* Swing
* Multi-Threading
* Buffered reader/writer
* File Input/output
* Collections
* JUnit

# **Description of Code**

## **Storage Location**

1. Unzip the folder Project.zip. The following files are stored under the folder,

Under Project

* ReadMe.docx

Under Project/SourceCode/businessLogic

* Sprinkler.java
* SprinklerConfigurationManager.java
* SprinklerEvent.java
* SprinklerSchedule.java

Under Project/SourceCode/userInterface

* BarChart.java
* GraphDemo.java
* ImagePanel.java
* SprinklerGUI.java

Under Project/SourceCode/dataStore

* SprinklerFileStore.java
* dailyConfigLogFile.txt
* graphWaterConsumption.txt
* graphweeklyWaterConsumption.txt
* weeklySchedule.txt
* weeklyWaterConsumption.txt

Under Project/SourceCode/juint

* SprinklerTestCases.java

Under Project/SourceCode/images

* + contains \*.png files used for the GUI display

Under folder project/DesignDocuments

* + Class diagram.png
  + Sequence diagrams.docx
  + State diagram.docx
  + UML diagrams.docx

Under folder project/JavaDocs

* Java Documents generated by using the plugin in eclipse

1. **Software Used**
2. Language Used: Java
3. Application: Eclipse
4. **Code Overview**
   1. Under Project/SourceCode/userInterface/ **SprinklerGUI.java** :

This file is used for the design of the sprinkler system user interface like welcomepage, status, configure, de/activate and settings

1. Initialize Method - sprinklerconfigmgr.initialize()
2. Used tabbed panels, for each functionality
3. Thread: Every second the sprinklerstatus is verified to repaint the image panel sprinklerconfigmgr.sprinklerStatus
4. Welcome page – welcomePage()
5. Status – statusPage()
   * 1. Activation Status - sprinklerconfigmgr.getGroupStatus(cmbType)
     2. Total Water consumed- sprinklerconfigmgr.getGroupWaterConsumption(cmbType)
     3. Graph - sprinklerconfigmgr.getGroupWaterConsumption("North")
6. Configure – configurePage()
   * 1. Apply Configuration - SprinklerConfigurationManager.*convertTimetoSec*(startTimeHours, startTimeMin, startTimeSec)
     2. sprinklerconfigmgr.applyDailyConfig(sprinklerID, cmbType, starttime, endtime, Integer.*parseInt*(temperature));
7. De/Activate – enablePage()
   * 1. sprinklerconfigmgr.handleEnableDisable(true, category, choice);
8. Settings – settings()
   * 1. Temperature - SprinklerConfigurationManager.*setCurrentTemperature*(systemTemperature);
     2. Break sprinkler - sprinklerconfigmgr.breakSprinkler(sprinklerID)
   1. Under Project/SourceCode/userInterface/ **ImagePanel.java** :

To display the garden layout along with the sprinklers configured on the right side of the panel

* 1. Under Project/SourceCode/userInterface/ **BarChart.java** :

To display bar chart based on the amount of water consumed by the garden sprinklers

* 1. Under Project/SourceCode/userInterface/ **GraphDemo.java** :

The volume of water consumed in each group are initialized and passed to the paint component in the bar chart, to paint the chart.

* 1. Under Project/SourceCode/businessLogic/ **SprinklerConfigurationManager.java**
  2. Implementation for temperature and sprinkler status methods[in Settings]
  3. Methods for activate/deactivate sprinklers [in De/Activate]
  4. Initialize methods sets the
     1. Weekly schedule for the sprinklers. So each time when the project is executed, it sets the values of the sprinkler by reading from the weekly schedule file
     2. Invokes the schedule worker thread. Thread: Schedule Worker Thread that is responsible for processing events and scheduling events based on sprinkler schedule
  5. Processes events by
     1. Process events at the start of sprinklerEventQueue
     2. Scan through sprinklerDaily to see the earliest nextEvent time
     3. Schedule Next Sprinkler Events
     4. Return least Next Event time
  6. Computes amount of water consumption
  7. Under Project/SourceCode/businessLogic/**Sprinkler.java:**

The sprinklers are configured with its values, ID’s and other initializations are done

* 1. Under Project/SourceCode/businessLogic/**SprinklerEvent.java**

The sprinkler event constructor and its get methods are declared

* 1. Under Project/SourceCode/businessLogic/**SprinklerSchedule.java**
     1. The temperature threshold is set to 40 for weekly schedule by default
     2. The new temperature change in the settings for the daily config are taken care
     3. The start and end time, get and set methods are declared
  2. Under Project/SourceCode/dataStore/**SprinklerFileStore.java**

Used buffered reader and writer to read/write the sprinkler schedules from the files

* 1. Under Project/SourceCode/dataStore/**dailyConfigLogFile.txt**

The new user configurations of the sprinklers are stored in the file

* 1. Under Project/SourceCode/dataStore/**graphWaterConsumption.txt**

The total water consumption of the each zones are stored in the file [Note: By default the formats are declared]

* 1. Under Project/SourceCode/dataStore/**graphweeklyWaterConsumption.txt**

The weekly water consumption of the each zones are stored in the file

* 1. Under Project/SourceCode/dataStore/**weeklySchedule.txt**

The default weekly schedules are stored in the file

* 1. Under Project/SourceCode/dataStore/**weeklyWaterConsumption.txt**

The weekly water consumption of the each zones are stored in the file

1. **How to execute code?**
2. Import project on eclipse in a project folder. Create separate packages like userInterface, businessLogic, dataStore and import respective into its package
3. To compile and execute

On eclipse

Run -> Run(Ctrl+F11)

1. **How to parse input?**
2. By default the sprinkler runs on the configuration as provided in the weeklyschedule.txt
3. If any new configuration is to be done for any sprinkler then the configure tab is used to set configuration for the respective sprinkler/zone
4. To activate/deactivate a zone or sprinkler the de/activate tab is used
5. Status of the sprinkler can be monitored and the graph for amount of water consumed can be viewed in the status tab
6. The sprinkler can be removed by using break and the temperature threshold can be changed by using the settings tab

# **Challenges**

The two big challenges faced are,

* The sprinklers are configured with the default weekly schedule configuration, but what happens when a new schedule is overlapped with the existing schedule?

For example: in case if sprinkler N1 on the north zone is configured to run from 9.56.01 to 10.56.01. If a new configuration by the user is N1 to 10.55.01 to 11.55.01 then, my design would schedule the N1 sprinkler from 9.56.01 to 11.55.01

* The sprinkler is configured to new schedule and the main thread does that, and the schedule worker thread computes the start/end time and configures the sprinkler. And at the same instance the sprinkler map worker thread repaints the map based the configuration of the sprinkler. So three threads are deployed.
* Supported four functionalities for a sprinkler

ON - activate - enable, functional, currently on

NOT ON - deactivate - disable - functional, currently not on

OK - functional

NOT OK - break - not functional

# **JUnit**

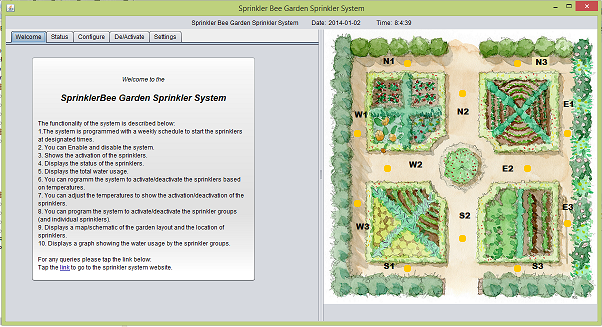
Implemented a Junit test framework, were all the individual test inputs are added to the test method to notify if there are any failures on the test inputs on the implemented code.

The test cases are geared towards testing the SprinkerConfigurationManager and Sprinkler class code, typically invoked by user GUI interactions, in isolation without invoking any of the GUI code.

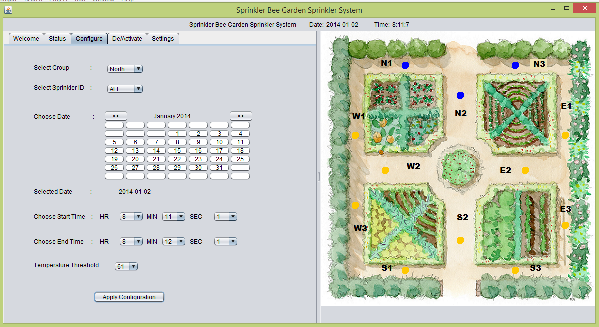
The run code on eclipse automatically runs the Junit Test. If all the test cases are passed, then it displays “0” errors/failures and nothing on the failure trace. If there is a failure, then it displays on the failure trace.

# **Screenshots**

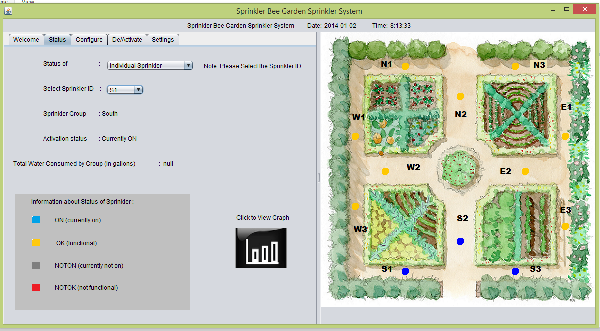
**Welcome Screen:**

****

**Configure Page:**

****

**Status**

****

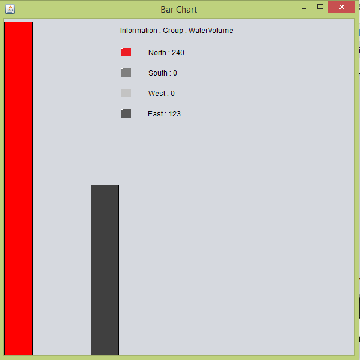
**De/Activate**

****

**Settings**

****

**Graph**

****